aws

User Guide

Amazon Simple Storage Service



API Version 2006-03-01

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Simple Storage Service: User Guide

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is Amazon S3? ... 1
Features of Amazon S3 ... 1
Storage classes ... 1
Storage management ... 2
Access management and security ... 3
Data processing ... 4
Storage logging and monitoring ... 4
Analytics and insights ... 5
Strong consistency ... 5
How Amazon S3 works ... 5
Buckets ... 6
Objects ... 9
Keys ... 9
S3 Versioning ... 10
Version ID ... 10
Bucket policy ... 10
S3 access points ... 10
Access control lists (ACLs) ... 11
Regions ... 11
Amazon S3 data consistency model ... 12
Concurrent applications ... 13
Related services ... 14
Accessing Amazon S3 ... 15
AWS Management Console ... 15
AWS Command Line Interface ... 15
AWS SDKs ... 15
Amazon S3 REST API ... 16
Paying for Amazon S3 ... 16
PCI DSS compliance ... 17
Amazon S3 Object Lambda availability change ... 18
Getting started ... 20
Setting up ... 20
Sign up for an AWS account ... 21
Step 1: Create a bucket ... 21

Step 2: Upload an object ... 27
Step 3: Download an object ... 28
Using the S3 console ... 28
Step 4: Copy an object ... 29
Step 5: Delete the objects and bucket ... 31
Deleting an object ... 31
Emptying your bucket ... 32
Deleting your bucket ... 32
Next steps ... 33
Understand common use cases ... 33
Control access to your buckets and objects ... 34
Protect and monitor your storage ... 35
Develop with Amazon S3 ... 35
Learn from tutorials ... 37
Explore training and support ... 38
Using Amazon S3 with the AWS CLI ... 38
Setting up ... 39
Step 1: Create your first Amazon S3 bucket ... 39
Step 2: Upload an object to your bucket ... 46
Step 3: Download an object ... 46
Step 4: Copy your object to a folder ... 48
Step 5: Delete your objects and bucket ... 48
Next steps ... 51
Working with general purpose buckets ... 53
General purpose buckets overview ... 54
General purpose buckets overview ... 55
Common general purpose bucket patterns ... 56
Permissions ... 56
Managing public access to general purpose buckets ... 57
Managing tags for general purpose buckets ... 58
General purpose buckets configuration ... 58
General purpose buckets operations ... 61
General purpose buckets performance monitoring ... 61
Namespaces for general purpose buckets ... 62
Global general purpose buckets ... 63
Account regional namespace general purpose buckets ... 63

Restrictions and considerations ... 66
AWS Region Code Format ... 66
Requiring the creation of buckets in your account regional namespace ... 67
Creating a bucket in your account regional namespace ... 69
Common bucket patterns ... 69
Multi-tenant general purpose bucket pattern ... 70
Bucket-per-use pattern ... 70
Naming rules ... 71
General purpose buckets naming rules ... 72
Account regional namespace naming rules ... 74
Example general purpose bucket names ... 75
Best practices ... 75
Creating a bucket that uses a GUID in the bucket name ... 77
Creating a bucket in your account regional namespace ... 78
Quotas, restrictions, and limitations ... 78
Bucket quotas ... 79
Objects and bucket limitations ... 79
Bucket naming rules ... 80
Accessing a bucket ... 80
Use cases ... 80
Amazon S3 console ... 82
AWS CLI ... 82
AWS SDKs ... 82
Amazon S3 REST API ... 83
Virtual hosting of general purpose buckets ... 83
Creating a general purpose bucket ... 92
General purpose bucket settings ... 93
Viewing bucket properties ... 105
Listing buckets ... 108
Emptying a general purpose bucket ... 112
Emptying a general purpose bucket with AWS CloudTrail configured ... 115
Deleting a general purpose bucket ... 115
Mountpoint for Amazon S3 ... 119
Installing Mountpoint ... 120
Configuring and using Mountpoint ... 126
Troubleshooting Mountpoint ... 131

Storage Browser for Amazon S3 ... 132
Using Storage Browser for S3 ... 133
Installing Storage Browser for S3 ... 134
Setting up Storage Browser for S3 ... 134
Configuring Storage Browser for S3 ... 148
Troubleshooting Storage Browser for S3 ... 149
Configuring Transfer Acceleration ... 149
Why use Transfer Acceleration? ... 150
Requirements for using Transfer Acceleration ... 150
Getting started ... 152
Enabling Transfer Acceleration ... 154
Speed Comparison tool ... 161
Using Requester Pays ... 162
How Requester Pays charges work ... 163
Configuring Requester Pays ... 164
Retrieving the requestPayment configuration ... 165
Downloading objects from Requester Pays buckets ... 166
Working with objects ... 168
Objects overview ... 169
Subresources ... 170
Naming objects ... 171
Choosing object key names ... 172
Object key naming guidelines ... 172
Object key sort order ... 177
Working with metadata ... 178
Custom object metadata ... 178
System-defined object metadata ... 181
Annotating objects ... 184
Tagging objects ... 195
Editing object metadata ... 207
Discovering your data with S3 Metadata tables ... 211
Uploading objects ... 320
Upload an object ... 322
Prevent uploading objects with identical key names ... 326
Using multipart upload ... 327
Making conditional requests ... 399

How to retrieve or copy objects based on metadata ... 400
How to prevent object overwrites ... 402
How to perform conditional deletes ... 415
Copying, moving, and renaming objects ... 420
Copying annotations ... 423
To copy an object ... 424
To move an object ... 433
To rename an object ... 434
Downloading objects ... 435
Downloading an object ... 437
Downloading multiple objects ... 438
Downloading part of an object ... 440
Downloading an object from another AWS account ... 441
Downloading archived objects ... 442
Downloading objects based on metadata ... 442
Troubleshooting downloading objects ... 443
Checking object integrity in Amazon S3 ... 443
Checking object integrity for data uploads in Amazon S3 ... 444
Checking object integrity for data at rest in Amazon S3 ... 466
Deleting objects ... 474
Best practices to consider before deleting an object ... 475
Deleting objects from a versioning-enabled bucket ... 476
Deleting objects from a versioning-suspended bucket ... 476
Deleting objects from an unversioned bucket ... 477
Deleting objects from an MFA-enabled bucket ... 477
Deleting a single object ... 478
Deleting multiple objects ... 487
Organizing and listing objects ... 490
Using prefixes ... 490
Listing objects ... 492
Using folders ... 495
Viewing object properties ... 500
Using presigned URLs to download and upload objects ... 502
Who can create a presigned URL ... 503
Expiration time for presigned URLs ... 504
Limiting presigned URL capabilities ... 504

Frequently asked questions for presigned URLs ... 506
Sharing objects with presigned URLs ... 507
Uploading objects with presigned URLs ... 512
Transforming objects ... 517
Creating Object Lambda Access Points ... 519
Using Amazon S3 Object Lambda Access Points ... 534
Security considerations ... 538
Writing Lambda functions ... 546
Using AWS built functions ... 578
Best practices and guidelines for S3 Object Lambda ... 581
S3 Object Lambda tutorials ... 583
Troubleshooting S3 Object Lambda ... 620
Performing object operations in bulk ... 621
Batch Operations basics ... 622
S3 Batch Operations tutorial ... 624
Granting permissions ... 624
Creating a job ... 638
Supported operations ... 658
Managing jobs ... 709
Tracking job status and completion reports ... 713
Using tags ... 728
Managing Object Lock with Batch Operations ... 742
Tutorial: Batch-transcoding videos ... 756
Troubleshooting S3 Batch Operations ... 796
Querying data in place ... 808
Requirements and limits ... 809
Constructing a request ... 809
Errors ... 811
S3 Select examples ... 811
SQL Reference ... 813
Working with directory buckets ... 853
Directory bucket names ... 854
Directories ... 854
Key names ... 854
Access management ... 855
Directory buckets quotas ... 855

Creating and using directory buckets ... 856
Use cases for directory buckets ... 856
High performance workloads ... 857
Data residency workloads ... 913
Differences for directory buckets ... 931
Differences for directory buckets ... 931
API operations supported for directory buckets ... 933
Amazon S3 features not supported by directory buckets ... 933
Networking for directory buckets ... 934
Endpoints ... 934
Configuring VPC gateway endpoints ... 935
Directory bucket naming rules ... 935
Viewing properties ... 936
Managing bucket policies ... 937
Adding a bucket policy ... 938
Viewing a bucket policy ... 941
Deleting a bucket policy ... 941
Emptying a directory bucket ... 942
Deleting a directory bucket ... 943
Listing directory buckets ... 946
Determining whether you can access a bucket ... 949
Working with objects in a directory bucket ... 950
Importing objects into a directory bucket ... 951
Working with S3 Lifecycle ... 952
Using Batch Operations with S3 Express One Zone ... 965
Appending data to objects ... 968
Renaming an object ... 971
Uploading an object ... 977
Copying an object ... 1009
Deleting an object ... 1015
Downloading an object ... 1019
Generating presigned URLs to share objects ... 1021
Retrieving object metadata ... 1022
Listing objects ... 1023
Security for directory buckets ... 1023
Data protection and encryption ... 1024

Authenticating and authorizing requests ... 1047
Security best practices ... 1060
Working with access points for directory buckets ... 1064
Naming rules, restrictions, and limitations ... 1065
Referencing access points for directory buckets ... 1067
Access points object operations ... 1067
Configuring IAM policies ... 1068
Monitoring and logging ... 1074
Creating access points for directory buckets ... 1075
Managing access points ... 1079
Tagging Access Points ... 1093
Logging with AWS CloudTrail for directory buckets ... 1108
CloudTrail management events for directory buckets ... 1109
CloudTrail data events for directory buckets ... 1109
... 1111
Monitoring metrics with CloudWatch for directory buckets ... 1118
Metrics and dimensions for directory buckets ... 1119
Configuring request metrics for directory buckets ... 1125
Optimizing directory bucket performance ... 1127
Use session-based authentication ... 1127
S3 additional checksum best practices ... 1127
Use the latest version of the AWS SDKs and common runtime libraries ... 1127
Developing with directory buckets ... 1128
Regional and Zonal endpoints for directory buckets ... 1128
Working with directory buckets by using the S3 console, AWS CLI, and AWS SDKs ... 1129
Directory bucket API operations ... 1131
Tagging directory buckets ... 1133
Common ways to use tags with directory buckets ... 1133
Managing tags for directory buckets ... 1137
Creating directory buckets with tags ... 1138
Adding a tag to a directory bucket ... 1141
Viewing directory bucket tags ... 1144
Deleting a tag from a directory bucket ... 1147
Resilience testing ... 1149
How it works ... 1150
Considerations and limitations ... 1150

Working with Amazon S3 Tables and table buckets 1152
Features of S3 Tables 1153
Related services 1154
Tutorial: Getting started with S3 Tables 1155
Step 1: Create a table bucket and integrate it with AWS analytics services 1156
Step 2: Create a table namespace and a table 1157
Step 3: Query data with SQL in Athena 1160
Table buckets 1161
Table bucket types 1161
Table bucket naming rules 1162
Create a table bucket 1164
Delete a table bucket 1167
Viewing table bucket details 1168
Managing policies 1169
AWS managed table buckets 1171
Tagging table buckets 1175
S3 Tables maintenance 1188
S3 Tables maintenance job status 1189
Table bucket maintenance 1190
Table maintenance 1192
Record expiration for tables 1198
Considerations and limitations 1204
S3 Tables Intelligent-Tiering 1207
S3 Tables Intelligent-Tiering access tiers 1208
Auto-tiering behavior with S3 Intelligent-Tiering 1208
Specifying S3 Intelligent-Tiering as your storage class 1209
Monitoring storage usage 1210
Namespaces 1212
Create a namespace 1213
Delete a namespace 1215
Tables 1215
Create a table 1217
Migrate a table 1223
Delete a table 1250
Viewing table details 1251
Managing policies 1253

Tagging tables ... 1255
Accessing tables ... 1268
Accessing tables through the AWS Glue Data Catalog integration ... 1268
Accessing tables directly ... 1269
S3 Tables integration overview ... 1270
Accessing tables using the AWS Glue Iceberg REST endpoint ... 1278
Accessing tables using the Amazon S3 Tables Iceberg REST endpoint ... 1283
Accessing tables with the client catalog ... 1289
Amazon Athena ... 1292
Amazon Redshift ... 1294
Amazon EMR ... 1295
Quick ... 1299
Amazon Data Firehose ... 1301
AWS Glue ETL ... 1307
Querying S3 Tables with SageMaker Unified Studio ... 1319
Streaming tables with Amazon MSK ... 1321
Working with Apache Iceberg V3 ... 1323
Key Features in V3 ... 1324
Version Compatibility ... 1325
Getting Started with V3 ... 1326
Best Practices for V3 ... 1333
Migration Strategy ... 1334
Compatibility Considerations ... 1334
Troubleshooting ... 1335
Pricing ... 1337
Availability ... 1337
Additional Resources ... 1337
Replicating S3 tables ... 1337
When to use S3 Tables replication ... 1338
Features ... 1338
How S3 Tables replication works ... 1339
Setting up S3 Tables replication ... 1343
Managing S3 Tables replication ... 1359
AWS Regions, endpoints, and quotas ... 1362
S3 Tables AWS Regions and endpoints ... 1362
S3 Tables quotas ... 1362

S3 Tables table migration quotas ... 1363
Making requests over IPv6 ... 1364
Getting started making S3 Tables requests over IPv6 ... 1365
Security for S3 Tables ... 1366
Encryption ... 1367
Access management ... 1381
VPC connectivity ... 1412
Restrictions and limitations ... 1418
Logging and monitoring ... 1418
Logging with AWS CloudTrail for S3 Tables ... 1419
Server access logging ... 1426
Monitoring metrics with Amazon CloudWatch ... 1443
Working with S3 Vectors and vector buckets ... 1452
What is Amazon S3 Vectors? ... 1452
Use cases: Similarity searches across large datasets ... 1452
Features of S3 Vectors ... 1453
Purpose-built storage for vectors ... 1453
Perform similarity queries ... 1454
Metadata filtering ... 1454
Access management and security ... 1454
Integration with AWS services ... 1454
Tutorial: Getting started with S3 Vectors ... 1455
Step 1: Create a vector bucket with the console ... 1455
Step 2: Create a vector index in a vector bucket with the console ... 1456
Step 3: Insert vectors into a vector index with the SDK for Python (Boto3) ... 1458
Step 4. Query vectors in a vector index with the SDK for Python (Boto3) ... 1460
(Optional) Automate vector embedding creation with the S3 Vectors Embed CLI ... 1462
(Optional) Integrate S3 Vectors with Amazon Bedrock Knowledge Bases ... 1462
(Optional) Integrate S3 Vectors with Amazon OpenSearch ... 1464
Vector buckets ... 1464
Vector bucket naming rules ... 1466
Creating a vector bucket ... 1466
Listing vector buckets ... 1469
Viewing vector bucket attributes ... 1471
Deleting a vector bucket ... 1471
Managing vector bucket policies ... 1473

Tagging vector buckets ... 1475
Vector indexes ... 1486
Vector index naming requirements ... 1486
Dimension requirements ... 1487
Distance metric options ... 1487
Non-filterable metadata keys ... 1487
Index mode ... 1488
Creating a vector index in a vector bucket ... 1489
Changing a vector index's mode ... 1494
Listing vector indexes ... 1496
Deleting a vector index ... 1499
Tagging vector indexes ... 1500
Vectors ... 1510
Vector concepts ... 1510
Inserting vectors into a vector index ... 1511
Listing vectors ... 1514
Querying vectors ... 1517
Deleting vectors from a vector index ... 1521
Metadata filtering ... 1521
Limitations and restrictions ... 1527
S3 Vectors best practices ... 1529
Creating and searching vector embeddings with `s3vectors-embed-cli` ... 1531
Using S3 Vectors with other AWS services ... 1532
Available integrations ... 1532
Integration benefits ... 1532
Using S3 Vectors with OpenSearch Service ... 1533
Using S3 Vectors with Amazon Bedrock Knowledge Bases ... 1536
AWS Regions, endpoints, and quotas for S3 Vectors ... 1540
S3 Vectors AWS Regions and endpoints ... 1540
S3 Vectors service quotas ... 1545
Security in S3 Vectors ... 1545
Identity and Access management in S3 Vectors ... 1546
Data protection and encryption in S3 Vectors ... 1562
VPC endpoints for S3 Vectors ... 1573
Logging with AWS CloudTrail for S3 Vectors ... 1579
S3 Vectors information in CloudTrail ... 1580

CloudTrail management events for S3 Vectors ... 1581
CloudTrail data events for S3 Vectors ... 1581
Enabling data event logging for S3 Vectors ... 1582
CloudTrail log file example for S3 Vectors ... 1583
Working with Amazon S3 Files ... 1586
What is S3 Files? ... 1586
How does S3 Files work? ... 1587
Are you a first-time user of S3 Files? ... 1588
Key concepts ... 1588
Features ... 1589
How are you billed for S3 Files? ... 1590
Prerequisites ... 1591
AWS account and compute setup ... 1591
S3 Files client ... 1591
IAM roles and policies ... 1592
Security groups ... 1598
Tutorial: Getting started with S3 Files ... 1598
Getting started with S3 Files using the AWS Console ... 1598
Getting started with S3 Files using the AWS CLI ... 1600
Mounting S3 buckets on compute resources ... 1603
Mounting on Amazon EC2 ... 1603
Mounting on AWS Lambda ... 1612
Mounting on Amazon EKS ... 1613
Mounting on Amazon ECS ... 1615
Cross-account mounting ... 1616
Creating and managing resources ... 1624
CloudFormation template ... 1625
Create file systems ... 1626
Delete file systems ... 1629
Create mount targets ... 1630
Manage mount targets ... 1631
Delete mount targets ... 1632
Creating file system policies ... 1633
Deleting file system policies ... 1633
Create access points ... 1634
Delete access points ... 1636

Tags ... 1637
Synchronizing data ... 1639
S3 bucket is accessible through the file system ... 1639
Changes in your file system automatically reflect in your S3 bucket ... 1640
Changes in your S3 bucket automatically appear in your file system ... 1640
Understanding the impact of rename and move operations ... 1641
Unused data is expired from the file system to optimize storage ... 1642
S3 bucket is the source of truth in case of conflicts ... 1642
Customize synchronization ... 1643
Monitoring and auditing ... 1646
CloudWatch monitoring ... 1647
CloudTrail logging ... 1651
Performance specifications ... 1657
Performance summary ... 1657
How S3 Files delivers performance ... 1657
Read performance ... 1658
Write performance ... 1658
First access latency ... 1658
Synchronization performance ... 1659
Monitoring performance ... 1660
Security ... 1660
Data Protection ... 1661
Encryption ... 1662
How S3 Files works with IAM ... 1665
Client access control examples ... 1670
POSIX permissions ... 1673
Metering ... 1678
How file system storage is metered ... 1678
How data access is metered ... 1679
How data access is metered from the file system ... 1679
How streaming directly from your S3 bucket is metered ... 1680
How bucket synchronization is metered ... 1680
Metering examples ... 1681
Best practices ... 1682
Performance and cost optimization ... 1682
Synchronization ... 1683

Access control ... 1683
Monitoring ... 1684
Migration ... 1684
Unsupported features, limits, and quotas ... 1684
Unsupported S3 features ... 1685
File system limitations ... 1685
Quotas ... 1685
Unsupported NFS features ... 1687
Troubleshooting ... 1688
Mount command fails ... 1688
Permission denied on file operations ... 1689
Intelligent read routing is not working ... 1689
File system consistently returns NFS server error ... 1690
Missing object in S3 bucket after file system write ... 1691
S3 object not visible in the file system ... 1692
Files appearing in the lost and found directory ... 1693
Synchronization falling behind ... 1694
File system status is MISCONFIGURED ... 1694
Enabling client debug logs ... 1695
Error messages ... 1697
Access control ... 1698
S3 resources ... 1699
Identities ... 1704
Bucket or resource owners ... 1706
Access management tools ... 1706
Actions ... 1712
Access management use cases ... 1713
Access management troubleshooting ... 1720
Identity and Access Management (IAM) ... 1721
Audience ... 1722
Authenticating with identities ... 1722
Managing access using policies ... 1724
How Amazon S3 works with IAM ... 1725
Request authorization ... 1751
Required permissions for S3 API operations ... 1759
Policies and permissions ... 1803

Bucket policies ... 1807
Identity-based policies ... 1860
Walkthroughs using policies ... 1898
Using service-linked roles for Amazon S3 Storage Lens ... 1938
Troubleshooting Amazon S3 identity and access ... 1942
AWS managed policies ... 1968
Working with access points ... 1970
Naming rules, restrictions, and limitations ... 1972
Referencing access points ... 1975
Access point compatibility ... 1979
Configuring IAM policies ... 1982
Monitoring and logging ... 1990
Creating an access point ... 1993
Managing access points ... 2000
Using access points ... 2005
Tagging Access Points ... 2017
Managing access with S3 Access Grants ... 2032
S3 Access Grants concepts ... 2034
S3 Access Grants and corporate directory identities ... 2038
Getting started with S3 Access Grants ... 2047
Working with S3 Access Grants instances ... 2049
Working with S3 Access Grants locations ... 2061
Working with grants in S3 Access Grants ... 2083
Getting S3 data using access grants ... 2097
S3 Access Grants cross-account access ... 2116
Managing tags for S3 Access Grants ... 2129
S3 Access Grants limitations ... 2131
S3 Access Grants integrations ... 2134
Managing access with ACLs ... 2136
ACL overview ... 2137
Configuring ACLs ... 2155
Policy examples ... 2172
Blocking public access ... 2177
Block public access settings ... 2179
Managing block public access at organization level ... 2183
Performing block public access operations on an access point ... 2184

The meaning of "public" ... 2184
Using IAM Access Analyzer for S3 to review public buckets ... 2187
Permissions ... 2188
Configuring block public access ... 2189
Configuring account settings ... 2189
Configuring bucket and access point settings ... 2192
Granting public access ... 2196
Granting public access to an entire bucket ... 2197
Granting public access to specific objects with object tags ... 2197
Granting public access to an individual object with an ACL ... 2198
Reviewing bucket access ... 2199
Reviewing external access with external access summary ... 2201
Information provided by IAM Access Analyzer for S3 ... 2203
Blocking all public access ... 2204
Reviewing and changing bucket access ... 2205
Archiving bucket findings ... 2206
Activating an archived bucket finding ... 2207
Viewing finding details ... 2207
Downloading an IAM Access Analyzer for S3 report ... 2208
Verifying bucket ownership ... 2208
When to use bucket owner condition ... 2209
Verifying a bucket owner ... 2210
Examples ... 2210
Restrictions and limitations ... 2213
Controlling object ownership ... 2213
Object Ownership settings ... 2215
Changes introduced by disabling ACLs ... 2217
Prerequisites for disabling ACLs ... 2219
Object Ownership permissions ... 2220
Disabling ACLs for all new buckets ... 2220
Replication and Object Ownership ... 2220
Setting Object Ownership ... 2220
Prerequisites for disabling ACLs ... 2221
Creating a bucket ... 2237
Setting Object Ownership ... 2245
Viewing Object Ownership settings ... 2249

Disabling ACLs for all new buckets ... 2250
Troubleshooting ... 2253
Security ... 2256
Security best practices ... 2258
Amazon S3 security best practices ... 2258
Amazon S3 monitoring and auditing best practices ... 2266
Monitoring data security ... 2269
Data protection ... 2272
Data encryption ... 2274
Server-side encryption ... 2276
Using client-side encryption ... 2392
Encryption for data in transit ... 2392
Internetwork traffic privacy ... 2399
Traffic between service and on-premises clients and applications ... 2399
Traffic between AWS resources in the same Region ... 2400
AWS PrivateLink for Amazon S3 ... 2400
Compliance validation ... 2421
Resilience ... 2423
Backup encryption ... 2425
Infrastructure security ... 2426
Configuration and vulnerability analysis ... 2427
Access management ... 2427
Amazon S3 data inventory ... 2429
Amazon S3 ... 2429
Geographic location ... 2430
Buckets ... 2431
Objects ... 2431
Storage Classes ... 2432
Storage management ... 2432
Access management and security ... 2433
Data Transfer ... 2433
Related Resources ... 2434
Data protection ... 2435
Replicating objects within and across Regions ... 2437
Why use replication? ... 2438
When to use Cross-Region Replication ... 2439

When to use Same-Region Replication ... 2440
When to use two-way replication (bi-directional replication) ... 2440
When to use S3 Batch Replication ... 2441
Workload requirements and live replication ... 2441
What's replicated? ... 2442
Requirements and considerations for replication ... 2446
Setting up live replication ... 2450
Managing or pausing live replication ... 2541
Replicating existing objects ... 2542
Troubleshooting replication ... 2557
Monitoring progress and getting status ... 2566
Managing multi-region traffic ... 2590
Creating Multi-Region Access Points ... 2592
Configuring Multi-Region Access Points ... 2602
Using Multi-Region Access Points ... 2608
Retaining multiple versions of objects ... 2661
Unversioned, versioning-enabled, and versioning-suspended buckets ... 2661
Using S3 Versioning with S3 Lifecycle ... 2662
S3 Versioning ... 2663
Enabling versioning on buckets ... 2667
Configuring MFA delete ... 2675
Working with versioning-enabled objects ... 2678
Working with versioning-suspended objects ... 2710
Troubleshooting versioning ... 2715
Locking objects ... 2719
How S3 Object Lock works ... 2721
Object Lock considerations ... 2728
Configuring Object Lock ... 2737
Backing up your data ... 2752
Cost optimization ... 2754
Billing and usage reporting ... 2755
Using cost allocation tags ... 2755
Billing reports ... 2758
Usage reports ... 2761
Understanding billing and usage reports ... 2763
Billing for Amazon S3 error responses ... 2792

Understanding and managing storage classes ... 2814
Frequently accessed objects ... 2815
Automatically optimizing data with changing or unknown access patterns ... 2816
Infrequently accessed objects ... 2818
Rarely accessed objects ... 2819
Amazon S3 on Outposts ... 2820
Comparing storage classes ... 2821
Setting the storage class of an object ... 2822
Storage Class Analysis ... 2826
Managing storage costs with Amazon S3 Intelligent-Tiering ... 2834
Amazon S3 Glacier storage classes ... 2846
Working with archived objects ... 2852
Managing lifecycle ... 2865
Managing the complete lifecycle of objects ... 2867
Transitioning objects ... 2868
Expiring objects ... 2876
Setting lifecycle configuration ... 2879
Using other bucket configurations ... 2896
Configuring S3 Lifecycle event notifications ... 2899
Lifecycle configuration elements ... 2901
Lifecycle configuration conflicts ... 2919
Examples of S3 Lifecycle configurations ... 2924
Troubleshooting lifecycle issues ... 2939
Logging and monitoring ... 2947
Monitoring tools ... 2950
Automated tools ... 2951
Manual tools ... 2951
Logging options ... 2952
Logging with CloudTrail ... 2955
Using CloudTrail logs with Amazon S3 server access logs and CloudWatch Logs ... 2957
CloudTrail tracking with Amazon S3 SOAP API calls ... 2957
CloudTrail events ... 2959
Example log files ... 2970
Enabling CloudTrail ... 2974
Identifying S3 requests ... 2977
Logging server access ... 2984

Delivery options ... 2985
Best-effort server log delivery ... 2986
Bucket logging status changes take effect over time ... 2987
Pricing ... 2987
Delivering to CloudWatch Logs ... 2987
Delivering to a GP bucket ... 3014
Monitoring metrics with CloudWatch ... 3060
Metrics and dimensions ... 3063
Accessing CloudWatch metrics ... 3081
CloudWatch metrics configurations ... 3083
Amazon S3 Event Notifications ... 3091
Overview ... 3092
Notification types and destinations ... 3094
Using SQS, SNS, and Lambda ... 3103
Using EventBridge ... 3132
Monitoring your storage activity and usage with S3 Storage Lens ... 3146
S3 Storage Lens metrics and features ... 3147
Understanding S3 Storage Lens ... 3150
Metrics glossary ... 3164
Setting permissions ... 3226
Working with S3 Storage Lens ... 3230
Viewing storage metrics ... 3272
Viewing metrics in AWS managed S3 Tables ... 3347
Working with Organizations ... 3389
Working with Storage Lens groups ... 3399
Cataloging and analyzing your data ... 3444
Amazon S3 Inventory buckets ... 3445
Inventory lists ... 3446
Configuring Amazon S3 Inventory ... 3452
Locating your inventory ... 3465
Setting up notifications for inventory completion ... 3469
Querying inventory with Athena ... 3470
Converting empty version ID strings to null strings ... 3476
Working with the Object ACL field ... 3479
Optimizing performance ... 3481
Performance guidelines for Amazon S3 ... 3482

Measure performance ... 3483
Scale horizontally ... 3483
Use byte-range fetches ... 3484
Retry requests ... 3484
Combine Amazon S3 and Amazon EC2 in the same Region ... 3484
Use Transfer Acceleration to minimize latency ... 3484
Use the latest AWS SDKs ... 3485
Performance design patterns for Amazon S3 ... 3485
Caching frequently accessed content ... 3486
Timeouts and retries for latency-sensitive apps ... 3486
Horizontal scaling and request parallelization ... 3487
Accelerating geographically disparate data transfers ... 3489
Optimizing for high-request rate workloads ... 3489
Hosting a static website ... 3491
Website endpoints ... 3492
Website endpoint examples ... 3494
Adding a DNS CNAME ... 3494
Using a custom domain with Route 53 ... 3495
Key differences between a website endpoint and a REST API endpoint ... 3495
Enabling website hosting ... 3496
Configuring an index document ... 3501
Index document and folders ... 3502
Configure an index document ... 3502
Configuring a custom error document ... 3504
Amazon S3 HTTP response codes ... 3505
Configuring a custom error document ... 3507
Setting permissions for website access ... 3508
Step 1: Edit S3 Block Public Access settings ... 3509
Step 2: Add a bucket policy ... 3510
Object access control lists ... 3512
Logging web traffic ... 3513
Configuring a redirect ... 3514
Redirect requests to another host ... 3514
Configure redirection rules ... 3515
Redirect requests for an object ... 3524
Using CORS ... 3526

Cross-origin resource sharing: Use-case scenarios ... 3526
How does Amazon S3 evaluate the CORS configuration on a bucket? ... 3527
How Object Lambda Access Point supports CORS ... 3527
Elements of a CORS configuration ... 3528
Configuring CORS ... 3533
Testing CORS ... 3542
Troubleshooting CORS ... 3544
Static website tutorials ... 3549
Hosting video streaming ... 3551
Configuring a static website ... 3570
Configuring a static website using a custom domain ... 3579
Deploying a static website to Amplify from Amazon S3 ... 3603
Tagging S3 resources ... 3607
How tags work ... 3607
User-defined tags ... 3607
Common ways to use tags ... 3609
Using tags for cost allocation ... 3610
Using tags for attribute-based access control (ABAC) ... 3611
Planning your tagging strategy ... 3612
Managing tags for Amazon S3 resources ... 3613
Tagging buckets ... 3614
Common ways to use tags with buckets ... 3614
ABAC for S3 general purpose buckets ... 3614
Example ABAC policies for buckets ... 3616
Managing tags for general purpose buckets ... 3618
Enable bucket ABAC ... 3618
Auditing your policies before enabling ABAC ... 3619
Creating general purpose buckets with tags ... 3623
Permissions ... 3624
Troubleshooting errors ... 3624
Steps ... 3624
Adding a tag to a bucket ... 3627
Permissions ... 3627
Troubleshooting errors ... 3627
Steps ... 3620
Viewing bucket tags ... 3629

Permissions 3629
Troubleshooting errors 3629
Steps 3629
Deleting a tag from a bucket 3632
Permissions 3632
Troubleshooting errors 3632
Steps 3632
Quotas 3635
Quota increases 3635
Reference 3636
Document history 3637
Earlier updates 3694

What is Amazon S3?

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. Customers of all sizes and industries can use Amazon S3 to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides management features so that you can optimize, organize, and configure access to your data to meet your specific business, organizational, and compliance requirements.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Topics

- Features of Amazon S3
- How Amazon S3 works
- Amazon S3 data consistency model
- Related services
- Accessing Amazon S3
- Paying for Amazon S3
- PCI DSS compliance

Features of Amazon S3

Storage classes

Amazon S3 offers a range of storage classes designed for different use cases. For example, you can store mission-critical production data in S3 Standard or S3 Express One Zone for frequent access, save costs by storing infrequently accessed data in S3 Standard-IA or S3 One Zone-IA, and archive data at the lowest costs in S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive.

Amazon S3 Express One Zone is a high-performance, single-zone Amazon S3 storage class that is purpose-built to deliver consistent, single-digit millisecond data access for your most latency-sensitive applications. S3 Express One Zone is the lowest latency cloud object storage class available today, with data access speeds up to 10x faster and with request costs 80 percent lower than S3 Standard. S3 Express One Zone is the first S3 storage class where you can select a single Availability Zone with the option to co-locate your object storage with your compute resources, which provides the highest possible access speed. Additionally, to further increase access speed and support hundreds of thousands of requests per second, data is stored in a new bucket type: an Amazon S3 directory bucket. For more information, see S3 Express One Zone and Working with directory buckets.

You can store data with changing or unknown access patterns in S3 Intelligent-Tiering, which optimizes storage costs by automatically moving your data between four access tiers when your access patterns change. These four access tiers include two low-latency access tiers optimized for frequent and infrequent access, and two opt-in archive access tiers designed for asynchronous access for rarely accessed data.

For more information, see Understanding and managing Amazon S3 storage classes.

Storage management

Amazon S3 has storage management features that you can use to manage costs, meet regulatory requirements, reduce latency, and save multiple distinct copies of your data for compliance requirements.

- S3 Lifecycle – Configure a lifecycle configuration to manage your objects and store them cost effectively throughout their lifecycle. You can transition objects to other S3 storage classes or expire objects that reach the end of their lifetimes.
- S3 Object Lock – Prevent Amazon S3 objects from being deleted or overwritten for a fixed amount of time or indefinitely. You can use Object Lock to help meet regulatory requirements that require *write-once-read-many (WORM)* storage or to simply add another layer of protection against object changes and deletions.
- S3 Replication – Replicate objects and their respective metadata and object tags to one or more destination buckets in the same or different AWS Regions for reduced latency, compliance, security, and other use cases.
- S3 Batch Operations – Manage billions of objects at scale with a single S3 API request or a few clicks in the Amazon S3 console. You can use Batch Operations to perform operations such as **Copy**, **Invoke AWS Lambda function**, and **Restore** on millions or billions of objects.

Access management and security

Amazon S3 provides features for auditing and managing access to your buckets and objects. By default, S3 buckets and the objects in them are private. You have access only to the S3 resources that you create. To grant granular resource permissions that support your specific use case or to audit the permissions of your Amazon S3 resources, you can use the following features.

- S3 Block Public Access – Block public access to S3 buckets and objects. By default, Block Public Access settings are turned on at the bucket level. We recommend that you keep all Block Public Access settings enabled unless you know that you need to turn off one or more of them for your specific use case. For more information, see Configuring block public access settings for your S3 buckets.
- AWS Identity and Access Management (IAM) – IAM is a web service that helps you securely control access to AWS resources, including your Amazon S3 resources. With IAM, you can centrally manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources.
- Bucket policies – Use IAM-based policy language to configure resource-based permissions for your S3 buckets and the objects in them.
- Amazon S3 access points – Configure named network endpoints with dedicated access policies to manage data access at scale for shared datasets in Amazon S3.
- Access control lists (ACLs) – Grant read and write permissions for individual buckets and objects to authorized users. As a general rule, we recommend using S3 resource-based policies (bucket policies and access point policies) or IAM user policies for access control instead of ACLs. Policies are a simplified and more flexible access control option. With bucket policies and access point policies, you can define rules that apply broadly across all requests to your Amazon S3 resources. For more information about the specific cases when you'd use ACLs instead of resource-based policies or IAM user policies, see Managing access with ACLs.
- S3 Object Ownership – Take ownership of every object in your bucket, simplifying access management for data stored in Amazon S3. S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to disable or enable ACLs. By default, ACLs are disabled. With ACLs disabled, the bucket owner owns all the objects in the bucket and manages access to data exclusively by using access-management policies.
- IAM Access Analyzer for S3 – Evaluate and monitor your S3 bucket access policies, ensuring that the policies provide only the intended access to your S3 resources.

Data processing

To transform data and trigger workflows to automate a variety of other processing activities at scale, you can use the following features.

- S3 Object Lambda – Add your own code to S3 GET, HEAD, and LIST requests to modify and process data as it is returned to an application. Filter rows, dynamically resize images, redact confidential data, and much more.
- Event notifications – Trigger workflows that use Amazon Simple Notification Service (Amazon SNS), Amazon Simple Queue Service (Amazon SQS), and AWS Lambda when a change is made to your S3 resources.

Storage logging and monitoring

Amazon S3 provides logging and monitoring tools that you can use to monitor and control how your Amazon S3 resources are being used. For more information, see Monitoring tools.

Automated monitoring tools

- Amazon CloudWatch metrics for Amazon S3 – Track the operational health of your S3 resources and configure billing alerts when estimated charges reach a user-defined threshold.
- AWS CloudTrail – Record actions taken by a user, a role, or an AWS service in Amazon S3. CloudTrail logs provide you with detailed API tracking for S3 bucket-level and object-level operations.

Manual monitoring tools

- Server access logging – Get detailed records for the requests that are made to a bucket. You can use server access logs for many use cases, such as conducting security and access audits, learning about your customer base, and understanding your Amazon S3 bill.
- AWS Trusted Advisor – Evaluate your account by using AWS best practice checks to identify ways to optimize your AWS infrastructure, improve security and performance, reduce costs, and monitor service quotas. You can then follow the recommendations to optimize your services and resources.

Analytics and insights

Amazon S3 offers features to help you gain visibility into your storage usage, which empowers you to better understand, analyze, and optimize your storage at scale.

- Amazon S3 Storage Lens – Understand, analyze, and optimize your storage. S3 Storage Lens provides 60+ usage and activity metrics and interactive dashboards to aggregate data for your entire organization, specific accounts, AWS Regions, buckets, or prefixes.
- Storage Class Analysis – Analyze storage access patterns to decide when it's time to move data to a more cost-effective storage class.
- S3 Inventory with Inventory reports – Audit and report on objects and their corresponding metadata and configure other Amazon S3 features to take action in Inventory reports. For example, you can report on the replication and encryption status of your objects. For a list of all the metadata available for each object in Inventory reports, see Amazon S3 Inventory list.

Strong consistency

Amazon S3 provides strong read-after-write consistency for PUT and DELETE requests of objects in your Amazon S3 bucket in all AWS Regions. This behavior applies to both writes of new objects as well as PUT requests that overwrite existing objects and DELETE requests. In addition, read operations on Amazon S3 Select, Amazon S3 access control lists (ACLs), Amazon S3 Object Tags, and object metadata (for example, the HEAD object) are strongly consistent. For more information, see Amazon S3 data consistency model.

How Amazon S3 works

Amazon S3 is an object storage service that stores data as objects, hierarchical data, or tabular data within buckets. An *object* is a file and any metadata that describes the file. A *bucket* is a container for objects.

To store your data in Amazon S3, you first create a bucket and specify a bucket name and AWS Region. Then, you upload your data to that bucket as objects in Amazon S3. Each object has a *key* (or *key name*), which is the unique identifier for the object within the bucket.

S3 provides features that you can configure to support your specific use case. For example, you can use S3 Versioning to keep multiple versions of an object in the same bucket, which allows you to restore objects that are accidentally deleted or overwritten.

Buckets and the objects in them are private and can be accessed only if you explicitly grant access permissions. You can use bucket policies, AWS Identity and Access Management (IAM) policies, access control lists (ACLs), and S3 Access Points to manage access.

Topics

- Buckets
- Objects
- Keys
- S3 Versioning
- Version ID
- Bucket policy
- S3 access points
- Access control lists (ACLs)
- Regions

Buckets

Amazon S3 supports four types of buckets—general purpose buckets, directory buckets, table buckets, and vector buckets. Each type of bucket provides a unique set of features for different use cases.

General purpose buckets – General purpose buckets are recommended for most use cases and access patterns and are the original S3 bucket type. A general purpose bucket is a container for objects stored in Amazon S3, and you can store any number of objects in a bucket and across all storage classes (except for S3 Express One Zone), so you can redundantly store objects across multiple Availability Zones. For more information, see Creating, configuring, and working with Amazon S3 general purpose buckets.

By default, general purpose buckets exist in a global namespace, which means that each bucket name must be unique across all AWS accounts in all the AWS Regions within a partition. A partition is a grouping of Regions. AWS currently has four partitions: `aws` (Standard Regions), `aws-cn` (China Regions), `aws-us-gov` (AWS GovCloud (US)), and `aws-eusc` (European Sovereign Cloud). When you create a general purpose bucket, you can choose to create a bucket in the shared global namespace or you can choose to create a bucket in your account regional namespace. Your account regional namespace is a subdivision of the global namespace that only your account can create buckets in. New general purpose buckets created in your account regional namespace are unique

to your account and can never be re-created by another account. For more information on bucket namespaces, see Namespaces for general purpose buckets.

> **Note**
>
> By default, all general purpose buckets are private. However, you can grant public access to general purpose buckets. You can control access to general purpose buckets at the bucket, prefix (folder), or object tag level. For more information, see Access control in Amazon S3.

Directory buckets – Recommended for low-latency use cases and data-residency use cases. By default, you can create up to 100 directory buckets in your AWS account, with no limit on the number of objects that you can store in a directory bucket. Directory buckets organize objects into hierarchical directories (prefixes) instead of the flat storage structure of general purpose buckets. This bucket type has no prefix limits and individual directories can scale horizontally. For more information, see Working with directory buckets.

- For low-latency use cases, you can create a directory bucket in a single AWS Availability Zone to store data. Directory buckets in Availability Zones support the S3 Express One Zone storage class. With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. The S3 Express One Zone storage class is recommended if your application is performance sensitive and benefits from single-digit millisecond PUT and GET latencies. To learn more about creating directory buckets in Availability Zones, see High performance workloads.
- For data-residency use cases, you can create a directory bucket in a single AWS Dedicated Local Zone (DLZ) to store data. In Dedicated Local Zones, you can create S3 directory buckets to store data in a specific data perimeter, which helps support your data residency and isolation use cases. Directory buckets in Local Zones support the S3 One Zone-Infrequent Access (S3 One Zone-IA; Z-IA) storage class. To learn more about creating directory buckets in Local Zones, see Data residency workloads.

> **Note**
>
> Directory buckets have all public access disabled by default. This behavior can't be changed. You can't grant access to objects stored in directory buckets. You can grant access only to your directory buckets. For more information, see Authenticating and authorizing requests.

Table buckets – Recommended for storing tabular data, such as daily purchase transactions, streaming sensor data, or ad impressions. Tabular data represents data in columns and rows, like in a database table. Table buckets provide S3 storage that's optimized for analytics and machine learning workloads, with features designed to continuously improve query performance and reduce storage costs for tables. S3 Tables are purpose-built for storing tabular data in the Apache Iceberg format. You can query tabular data in S3 Tables with popular query engines, including Amazon Athena, Amazon Redshift, and Apache Spark. By default, you can create up to 10 table buckets per AWS account per AWS Region and up to 10,000 tables per table bucket. For more information, see Working with S3 Tables and table buckets.

> **Note**
>
> All table buckets and tables are private and can't be made public. These resources can only be accessed by users who are explicitly granted access. To grant access, you can use IAM resource-based policies for table buckets and tables, and IAM identity-based policies for users and roles. For more information, see Security for S3 Tables.

Vector buckets – S3 vector buckets are a type of Amazon S3 bucket that are purpose-built to store and query vectors. Vector buckets use dedicated API operations to write and query vector data efficiently. With S3 vector buckets, you can store vector embeddings for machine learning models, perform similarity searches, and integrate with services like Amazon Bedrock and Amazon OpenSearch.

S3 vector buckets organize data using vector indexes, which are resources within a bucket that store and organize vector data for efficient similarity search. Each vector index can be configured with specific dimensions, distance metrics (like cosine similarity), and metadata configurations to optimize for your specific use case. For more information, see Working with S3 Vectors and vector buckets.

Additional information about all bucket types

When you create a bucket, you enter a bucket name and choose the AWS Region where the bucket will reside. After you create a bucket, you cannot change the name of the bucket or its Region. Bucket names must follow the following bucket naming rules:

- General purpose bucket naming rules
- Directory bucket naming rules

- Table bucket naming rules

Buckets also:

- Organize the Amazon S3 namespace at the highest level. For general purpose buckets, this namespace is S3. For directory buckets, this namespace is `s3express`. For table buckets, this namespace is `s3tables`.
- Identify the account responsible for storage and data transfer charges.
- Serve as the unit of aggregation for usage reporting.

Objects

Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata. The metadata is a set of name-value pairs that describe the object. These pairs include some default metadata, such as the date last modified, and standard HTTP metadata, such as `Content-Type`. You can also specify custom metadata at the time that the object is stored.

Every object is contained in a bucket. For example, if the object named `photos/puppy.jpg` is stored in the `amzn-s3-demo-bucket` general purpose bucket in the US West (Oregon) Region, then it is addressable by using the URL `https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg`. For more information, see Accessing a Bucket.

An object is uniquely identified within a bucket by a key (name) and a version ID (if S3 Versioning is enabled on the bucket). For more information about objects, see Amazon S3 objects overview.

Keys

An *object key* (or *key name*) is the unique identifier for an object within a bucket. Every object in a bucket has exactly one key. The combination of a bucket, object key, and optionally, version ID (if S3 Versioning is enabled for the bucket) uniquely identify each object. So you can think of Amazon S3 as a basic data map between "bucket + key + version" and the object itself.

Every object in Amazon S3 can be uniquely addressed through the combination of the web service endpoint, bucket name, key, and optionally, a version. For example, in the URL `https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg`, *`amzn-s3-demo-bucket`* is the name of the bucket and `photos/puppy.jpg` is the key.

For more information about object keys, see Naming Amazon S3 objects.

S3 Versioning

You can use S3 Versioning to keep multiple variants of an object in the same bucket. With S3 Versioning, you can preserve, retrieve, and restore every version of every object stored in your buckets. You can easily recover from both unintended user actions and application failures.

For more information, see Retaining multiple versions of objects with S3 Versioning.

Version ID

When you enable S3 Versioning in a bucket, Amazon S3 generates a unique version ID for each object added to the bucket. Objects that already existed in the bucket at the time that you enable versioning have a version ID of `null`. If you modify these (or any other) objects with other operations, such as CopyObject and PutObject, the new objects get a unique version ID.

For more information, see Retaining multiple versions of objects with S3 Versioning.

Bucket policy

A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. Bucket policies are limited to 20 KB in size.

Bucket policies use JSON-based access policy language that is standard across AWS. You can use bucket policies to add or deny permissions for the objects in a bucket. Bucket policies allow or deny requests based on the elements in the policy, including the requester, S3 actions, resources, and aspects or conditions of the request (for example, the IP address used to make the request). For example, you can create a bucket policy that grants cross-account permissions to upload objects to an S3 bucket while ensuring that the bucket owner has full control of the uploaded objects. For more information, see Examples of Amazon S3 bucket policies.

In your bucket policy, you can use wildcard characters on Amazon Resource Names (ARNs) and other values to grant permissions to a subset of objects. For example, you can control access to groups of objects that begin with a common prefix or end with a given extension, such as `.html`.

S3 access points

Amazon S3 access points are named network endpoints with dedicated access policies that describe how data can be accessed using that endpoint. Access points are attached to an

underlying data source, such as a general purpose bucket, directory bucket, or a FSx for OpenZFS volume, that you can use to perform S3 object operations, such as `GetObject` and `PutObject`. Access points simplify managing data access at scale for shared datasets in Amazon S3.

Each access point has its own access point policy. You can configure Block Public Access settings for each access point attached to a bucket. To restrict Amazon S3 data access to a private network, you can also configure any access point to accept requests only from a virtual private cloud (VPC).

For more information about access points for general purpose buckets, see Managing access to shared datasets with access points. For more information about access points for directory buckets, see Managing access to shared datasets in directory buckets with access points.

Access control lists (ACLs)

You can use ACLs to grant read and write permissions to authorized users for individual general purpose buckets and objects. Each general purpose bucket and object has an ACL attached to it as a subresource. The ACL defines which AWS accounts or groups are granted access and the type of access. ACLs are an access control mechanism that predates IAM. For more information about ACLs, see Access control list (ACL) overview.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

Regions

You can choose the geographical AWS Region where Amazon S3 stores the buckets that you create. You might choose a Region to optimize latency, minimize costs, or address regulatory requirements. Objects stored in an AWS Region never leave the Region unless you explicitly transfer or replicate them to another Region. For example, objects stored in the Europe (Ireland) Region never leave it.

> **Note**
>
> You can access Amazon S3 and its features only in the AWS Regions that are enabled for your account. For more information about enabling a Region to create and manage AWS resources, see Managing AWS Regions in the *AWS General Reference*.

For a list of Amazon S3 Regions and endpoints, see Regions and endpoints in the *AWS General Reference*.

Amazon S3 data consistency model

Amazon S3 provides strong read-after-write consistency for PUT and DELETE requests of objects in your Amazon S3 bucket in all AWS Regions. This behavior applies to both writes to new objects as well as PUT requests that overwrite existing objects and DELETE requests. In addition, read operations on Amazon S3 Select, Amazon S3 access controls lists (ACLs), Amazon S3 Object Tags, and object metadata (for example, the HEAD object) are strongly consistent.

Updates to a single key are atomic. For example, if you make a PUT request to an existing key from one thread and perform a GET request on the same key from a second thread concurrently, you will get either the old data or the new data, but never partial or corrupt data.

Amazon S3 achieves high availability by replicating data across multiple servers within AWS data centers. If a PUT request is successful, your data is safely stored. Any read (GET or LIST request) that is initiated following the receipt of a successful PUT response will return the data written by the PUT request. Here are examples of this behavior:

- A process writes a new object to Amazon S3 and immediately lists keys within its bucket. The new object appears in the list.
- A process replaces an existing object and immediately tries to read it. Amazon S3 returns the new data.
- A process deletes an existing object and immediately tries to read it. Amazon S3 does not return any data because the object has been deleted.
- A process deletes an existing object and immediately lists keys within its bucket. The object does not appear in the listing.

Note

- Amazon S3 does not support object locking for concurrent writers. If two PUT requests are simultaneously made to the same key, the request with the latest timestamp wins. If this is an issue, you must build an object-locking mechanism into your application.
- Updates are key-based. There is no way to make atomic updates across keys. For example, you cannot make the update of one key dependent on the update of another key unless you design this functionality into your application.

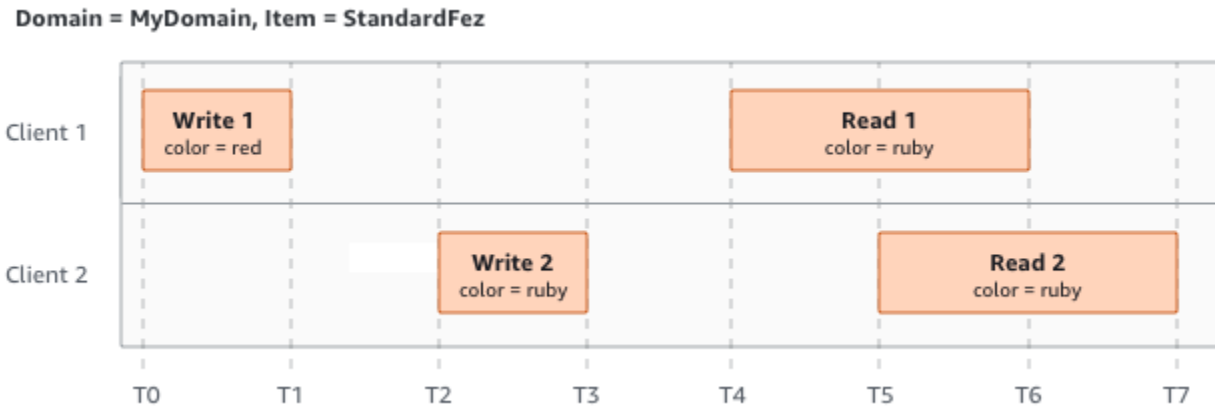
Bucket configurations have an eventual consistency model. Specifically, this means that:

- If you delete a bucket and immediately list all buckets, the deleted bucket might still appear in the list.
- If you enable versioning on a bucket for the first time, it might take a short amount of time for the change to be fully propagated. We recommend that you wait for 15 minutes after enabling versioning before issuing write operations (PUT or DELETE requests) on objects in the bucket.

Concurrent applications

This section provides examples of behavior to be expected from Amazon S3 when multiple clients are writing to the same items.

In this example, both W1 (write 1) and W2 (write 2) finish before the start of R1 (read 1) and R2 (read 2). Because S3 is strongly consistent, R1 and R2 both return `color = ruby`.



In the next example, W2 does not finish before the start of R1. Therefore, R1 might return `color = ruby` or `color = garnet`. However, because W1 and W2 finish before the start of R2, R2 returns `color = garnet`.



In the last example, W2 begins before W1 has received an acknowledgment. Therefore, these writes are considered concurrent. Amazon S3 internally uses last-writer-wins semantics to determine which write takes precedence. However, the order in which Amazon S3 receives the requests and the order in which applications receive acknowledgments cannot be predicted because of various factors, such as network latency. For example, W2 might be initiated by an Amazon EC2 instance in the same Region, while W1 might be initiated by a host that is farther away. The best way to determine the final value is to perform a read after both writes have been acknowledged.



Related services

After you load your data into Amazon S3, you can use it with other AWS services. The following are the services that you might use most frequently:

- **Amazon Elastic Compute Cloud (Amazon EC2)** – Provides secure and scalable computing capacity in the AWS Cloud. Using Amazon EC2 eliminates your need to invest in hardware upfront, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.
- **Amazon EMR** – Helps businesses, researchers, data analysts, and developers easily and cost-effectively process vast amounts of data. Amazon EMR uses a hosted Hadoop framework running on the web-scale infrastructure of Amazon EC2 and Amazon S3.
- **AWS Transfer Family** – Provides fully managed support for file transfers directly into and out of Amazon S3 or Amazon Elastic File System (Amazon EFS) using Secure Shell (SSH) File Transfer Protocol (SFTP), File Transfer Protocol over SSL (FTPS), and File Transfer Protocol (FTP).

Accessing Amazon S3

You can work with Amazon S3 in any of the following ways:

AWS Management Console

The console is a web-based user interface for managing Amazon S3 and AWS resources. If you've signed up for an AWS account, you can access the Amazon S3 console by signing into the AWS Management Console and choosing **S3** from the AWS Management Console home page.

AWS Command Line Interface

You can use the AWS command line tools to issue commands or build scripts at your system's command line to perform AWS (including S3) tasks.

The AWS Command Line Interface (AWS CLI) provides commands for a broad set of AWS services. The AWS CLI is supported on Windows, macOS, and Linux. To get started, see the *AWS Command Line Interface User Guide*. For more information about the commands for Amazon S3, see s3api and s3control in the *AWS CLI Command Reference*.

AWS SDKs

AWS provides SDKs (software development kits) that consist of libraries and sample code for various programming languages and platforms (Java, Python, Ruby, .NET, iOS, Android, and so on). The AWS SDKs provide a convenient way to create programmatic access to S3 and AWS. Amazon S3 is a REST service. You can send requests to Amazon S3 using the AWS SDK libraries, which wrap

the underlying Amazon S3 REST API and simplify your programming tasks. For example, the SDKs take care of tasks such as calculating signatures, cryptographically signing requests, managing errors, and retrying requests automatically. For information about the AWS SDKs, including how to download and install them, see Tools for AWS.

Every interaction with Amazon S3 is either authenticated or anonymous. If you are using the AWS SDKs, the libraries compute the signature for authentication from the keys that you provide. For more information about how to make requests to Amazon S3, see Making requests .

Amazon S3 REST API

The architecture of Amazon S3 is designed to be programming language-neutral, using AWS-supported interfaces to store and retrieve objects. You can access S3 and AWS programmatically by using the Amazon S3 REST API. The REST API is an HTTP interface to Amazon S3. With the REST API, you use standard HTTP requests to create, fetch, and delete buckets and objects.

To use the REST API, you can use any toolkit that supports HTTP. You can even use a browser to fetch objects, as long as they are anonymously readable.

The REST API uses standard HTTP headers and status codes, so that standard browsers and toolkits work as expected. In some areas, we have added functionality to HTTP (for example, we added headers to support access control). In these cases, we have done our best to add the new functionality in a way that matches the style of standard HTTP usage.

If you make direct REST API calls in your application, you must write the code to compute the signature and add it to the request. For more information about how to make requests to Amazon S3, see Making requests in the *Amazon S3 API Reference*.

> **Note**
>
> SOAP API support over HTTP is deprecated, but it is still available over HTTPS. Newer Amazon S3 features are not supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

Paying for Amazon S3

Pricing for Amazon S3 is designed so that you don't have to plan for the storage requirements of your application. Most storage providers require you to purchase a predetermined amount of

storage and network transfer capacity. In this scenario, if you exceed that capacity, your service is shut off or you are charged high overage fees. If you do not exceed that capacity, you pay as though you used it all.

Amazon S3 charges you only for what you actually use, with no hidden fees and no overage charges. This model gives you a variable-cost service that can grow with your business while giving you the cost advantages of the AWS infrastructure. For more information, see Amazon S3 Pricing.

When you sign up for AWS, your AWS account is automatically signed up for all services in AWS, including Amazon S3. However, you are charged only for the services that you use. If you are a new Amazon S3 customer, you can get started with Amazon S3 for free. For more information, see AWS free tier.

To see your bill, go to the Billing and Cost Management Dashboard in the AWS Billing and Cost Management console. To learn more about AWS account billing, see the *AWS Billing User Guide*. If you have questions concerning AWS billing and AWS accounts, contact AWS Support.

PCI DSS compliance

Amazon S3 supports the processing, storage, and transmission of credit card data by a merchant or service provider, and has been validated as being compliant with Payment Card Industry (PCI) Data Security Standard (DSS). For more information about PCI DSS, including how to request a copy of the AWS PCI Compliance Package, see PCI DSS Level 1.

Amazon S3 Object Lambda availability change

After careful consideration, S3 Object Lambda as of November 7th, 2025 is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. Existing customers of S3 Object Lambda, as well as customers using or deploying APN partner solutions, can continue to use the service as usual. AWS will prioritize security and availability improvements for S3 Object Lambda, but we do not plan to introduce new capabilities

There are several alternative services and designs that allow you to modify and process data accessed from S3 to meet the needs of various client applications or users who are accessing the data. Those alternatives include the AWS Solution Dynamic Image Transformation for Amazon CloudFront, invoking AWS Lambda by other means (through CloudFront, API Gateway, or function URLs), or processing data in the client application. All of these alternatives continue to use Amazon S3 for the underlying storage, so no data migration is necessary.

In this post, we discuss how to choose the best option for your use case.

Dynamic Image Transformation for Amazon CloudFront

This AWS solution enables real-time image processing through the global content delivery network (CDN) of Amazon CloudFront using API Gateway and Lambda. It supports a variety of transformations such as format changes, dimensions, fit methods, rotations, and filters. If you are using S3 Object Lambda today for image transformation, this can be a good alternative. The solution overview has more details on capabilities and how to get started using it. Be sure to modify the Enable S3 Object Lambda template parameter to "No" when deploying the solution to your account.

For more information, see Dynamic Image Transformation for Amazon CloudFront.

Data processing in AWS Lambda

You can continue to use AWS Lambda for your data processing, but invoke it directly or through other AWS services. Your data processing logic can remain the same, but your Lambda function will need to be updated based on how you choose to invoke it. This option is best for use cases such as sensitive data redaction, format changes, or situations where the application that invokes S3 Object Lambda relies on receiving processed data. This option also minimizes the changes required if you choose to migrate off S3 Object Lambda.

If you are using an S3 Object Lambda access point as the origin of a CloudFront distribution, you can create a new origin using either Lambda Function URLs or API Gateway. This blog details how you can setup a Lambda Function URL to act as an origin for your CloudFront distribution.

If you directly invoke S3 Object Lambda from your client code today, you can use Lambda Function URLs to directly invoke your Lambda function or you can use API Gateway. If you're not sure which is the best method for your use case, see Select a method to invoke your Lambda function using an HTTP request in the *AWS Lambda Developer Guide*. After you have decided which method for invoking Lambda is best for your use case and have setup your Lambda function accordingly, you will also need to update your calling application to invoke your Lambda function rather than calling S3 Object Lambda.

Data processing in the client application

You also have the option of moving your data processing logic from S3 Object Lambda into your client application. This works best if you are using S3 Object Lambda as part of an application that already performs further processing or analysis of the data returned by S3 Object Lambda. For example, if your S3 Object Lambda access point was responsible for redirecting to particular objects or object versions, that redirect logic can be moved into the calling application which would then directly access data in the S3 bucket.

If you need assistance or have feedback, contact AWS Support.

Getting started with Amazon S3

You can get started with Amazon S3 by working with buckets and objects. A *bucket* is a container for objects. An *object* is a file and any metadata that describes that file.

To store an object in Amazon S3, you create a bucket and then upload the object to the bucket. When the object is in the bucket, you can open it, download it, and move it. When you no longer need an object or a bucket, you can clean up your resources.

With Amazon S3, you pay only for what you use. For more information about Amazon S3 features and pricing, see Amazon S3. If you are a new Amazon S3 customer, you can get started with Amazon S3 for free. For more information, see AWS Free Tier.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Video: Getting started with Amazon S3

Watch this

Prerequisites

Before you begin, confirm that you've completed the steps in Setting up Amazon S3.

Setting up Amazon S3

When you sign up for AWS, your AWS account is automatically signed up for all services in AWS, including Amazon S3. You are charged only for the services that you use.

With Amazon S3, you pay only for what you use. For more information about Amazon S3 features and pricing, see Amazon S3. If you are a new Amazon S3 customer, you can get started with Amazon S3 for free. For more information, see AWS Free Tier.

To set up Amazon S3, use the steps in the following sections.

When you sign up for AWS and set up Amazon S3, you can optionally change the display language in the AWS Management Console. For more information, see Changing the language of the AWS Management Console in the *AWS Management Console Getting Started Guide*.

Topics

- Sign up for an AWS account

Sign up for an AWS account

To get started with AWS, you need an AWS account. There are two ways to sign up for AWS. If you want complete control over account configuration and where your content is stored, or if you have regulated workloads such as HIPAA or FedRAMP, use Sign up for AWS (advanced). If you are a new user and want to use a login you already own to sign in to AWS and create an AWS environment with preconfigured defaults in select AWS Regions, use Sign up for AWS.

Step 1: Create your first S3 bucket

After you sign up for AWS, you're ready to create a bucket in Amazon S3 using the AWS Management Console. Every object in Amazon S3 is stored in a *bucket*. Before you can store data in Amazon S3, you must create a bucket.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

> **Note**
>
> You are not charged for creating a bucket. You are charged only for storing objects in the bucket and for transferring objects in and out of the bucket. The charges that you incur through following the examples in this guide are minimal (less than $1). For more information about storage charges, see Amazon S3 pricing.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

   Note

   - After you create a bucket, you can't change its Region.
   - To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.

3. In the left navigation pane, choose **General purpose buckets**.
4. Choose **Create bucket**. The **Create bucket** page opens.
5. For **Bucket name**, enter a name for your bucket.

   The bucket name must:

   - Be unique within a partition. A partition is a grouping of Regions. AWS currently has three partitions: `aws` (commercial Regions), `aws-cn` (China Regions), and `aws-us-gov` (AWS GovCloud (US) Regions).
   - Be between 3 and 63 characters long.
   - Consist only of lowercase letters, numbers, periods (`.`), and hyphens (`-`). For best compatibility, we recommend that you avoid using periods (`.`) in bucket names, except for buckets that are used only for static website hosting.
   - Begin and end with a letter or number.
   - For a complete list of bucket-naming rules, see General purpose bucket naming rules.

   Important

   - After you create the bucket, you can't change its name.
   - Don't include sensitive information in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

6. (Optional) Under **General configuration**, you can choose to copy an existing bucket's settings to your new bucket. If you don't want to copy the settings of an existing bucket, skip to the next step.

   > **Note**
   >
   > This option:
   >
   > - Isn't available in the AWS CLI and is only available in the Amazon S3 console
   > - Doesn't copy the bucket policy from the existing bucket to the new bucket

   To copy an existing bucket's settings, under **Copy settings from existing bucket**, select **Choose bucket**. The **Choose bucket** window opens. Find the bucket with the settings that you want to copy, and select **Choose bucket**. The **Choose bucket** window closes, and the **Create bucket** window reopens.

   Under **Copy settings from existing bucket**, you now see the name of the bucket that you selected. The settings of your new bucket now match the settings of the bucket that you selected. If you want to remove the copied settings, choose **Restore defaults**. Review the remaining bucket settings on the **Create bucket** page. If you don't want to make any changes, you can skip to the final step.

7. Under **Object Ownership**, to disable or enable ACLs and control ownership of objects uploaded in your bucket, choose one of the following settings:

   ACLs disabled

   - **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the general purpose bucket. ACLs no longer affect access permissions to data in the S3 general purpose bucket. The bucket uses policies exclusively to define access control.

     By default, ACLs are disabled. A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in circumstances where you must control access for each object individually. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

ACLs enabled

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.

  If you apply the **Bucket owner preferred** setting, to require all Amazon S3 uploads to include the `bucket-owner-full-control` canned ACL, you can add a bucket policy that allows only object uploads that use this ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

> **Note**
>
> The default setting is **Bucket owner enforced**. To apply the default setting and keep ACLs disabled, only the `s3:CreateBucket` permission is needed. To enable ACLs, you must have the `s3:PutBucketOwnershipControls` permission.

8. Under **Block Public Access settings for this bucket**, choose the Block Public Access settings that you want to apply to the bucket.

   By default, all four Block Public Access settings are enabled. We recommend that you keep all settings enabled, unless you know that you need to turn off one or more of them for your specific use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

   > **Note**
   >
   > To enable all Block Public Access settings, only the `s3:CreateBucket` permission is required. To turn off any Block Public Access settings, you must have the `s3:PutBucketPublicAccessBlock` permission.

9. (Optional) By default, **Bucket Versioning** is disabled. Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your bucket. With versioning, you can recover more easily from both unintended user actions and application failures. For more information about versioning, see Retaining multiple versions of objects with S3 Versioning.

To enable versioning on your bucket, choose **Enable**.

10. (Optional) Under **Tags**, you can choose to add tags to your bucket. With AWS cost allocation, you can use bucket tags to annotate billing for your use of a bucket. A tag is a key-value pair that represents a label that you assign to a bucket. For more information, see the section called "Using cost allocation tags".

    To add a bucket tag, enter a **Key** and optionally a **Value** and choose **Add Tag**.

11. To configure **Default encryption**, under **Encryption type**, choose one of the following:

    - **Server-side encryption with Amazon S3 managed keys (SSE-S3)**
    - **Server-side encryption with AWS Key Management Service keys (SSE-KMS)**
    - **Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)**

    > **Important**
    >
    > If you use the SSE-KMS or DSSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quota of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.

    Buckets and new objects are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption configuration. For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets. For more information about SSE-S3, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

    For more information about using server-side encryption to encrypt your data, see the section called "Data encryption".

12. If you chose **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** or **Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)**, do the following:

    a. Under **AWS KMS key**, specify your KMS key in one of the following ways:

- To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

  Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
- To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that isn't listed, you must enter your KMS key ARN. If you want to use a KMS key that's owned by a different account, you must first have permission to use the key, and then you must enter the KMS key ARN. For more information about cross account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*. For more information about SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS). For more information about DSSE-KMS, see the section called "Dual-layer server-side encryption (DSSE-KMS)". When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

b. When you configure your bucket to use default encryption with SSE-KMS, you can also use S3 Bucket Keys. S3 Bucket Keys lower the cost of encryption by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys. S3 Bucket Keys aren't supported for DSSE-KMS.

By default, S3 Bucket Keys are enabled in the Amazon S3 console. We recommend leaving S3 Bucket Keys enabled to lower your costs. To disable S3 Bucket Keys for your bucket, under **Bucket Key**, choose **Disable**.

13. (Optional) S3 Object Lock helps protect new objects from being deleted or overwritten. For more information, see Locking objects with Object Lock. If you want to enable S3 Object Lock, do the following:

    a. Choose **Advanced settings**.

    > **Important**
    >
    > Enabling Object Lock automatically enables versioning for the bucket. After you've enabled and successfully created the bucket, you must also configure the Object Lock default retention and legal hold settings on the bucket's **Properties** tab.

    b. If you want to enable Object Lock, choose **Enable**, read the warning that appears, and acknowledge it.

> **Note**
>
> To create an Object Lock enabled bucket, you must have the following permissions: `s3:CreateBucket`, `s3:PutBucketVersioning`, and `s3:PutBucketObjectLockConfiguration`.

14. Choose **Create bucket**.

You've created a bucket in Amazon S3.

Next step

To add an object to your bucket, see Step 2: Upload an object to your bucket.

Step 2: Upload an object to your bucket

After creating a bucket in Amazon S3, you're ready to upload an object to the bucket. An object can be any kind of file: a text file, a photo, a video, and so on.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

To upload an object to a bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to upload your object to.
3. On the **Objects** tab for your bucket, choose **Upload**.
4. Under **Files and folders**, choose **Add files**.
5. Choose a file to upload, and then choose **Open.**
6. Choose **Upload**.

You've successfully uploaded an object to your bucket.

Next step

To view your object, see Step 3: Download an object.

Step 3: Download an object

After you upload an object to a bucket, you can view information about your object and download the object to your local computer.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Using the S3 console

This section explains how to use the Amazon S3 console to download an object from an S3 bucket.

Note

- You can download only one object at a time.
- If you use the Amazon S3 console to download an object whose key name ends with a period (`.`), the period is removed from the key name of the downloaded object. To retain the period at the end of the name of the downloaded object, you must use the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

To download an object from an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets** or **Directory buckets**.
3. In the buckets list, choose the name of the bucket that you want to download an object from.
4. You can download an object from an S3 bucket in any of the following ways:
   - Select the check box next to the object, and choose **Download**. If you want to download the object to a specific folder, on the **Actions** menu, choose **Download as**.
   - If you want to download a specific version of the object, turn on **Show versions** (located next to the search box). Select the check box next to the version of the object that you want, and choose **Download**. If you want to download the object to a specific folder, on the **Actions** menu, choose **Download as**.

You've successfully downloaded your object.

Next step

To copy and paste your object within Amazon S3, see Step 4: Copy your object to a folder.

Step 4: Copy your object to a folder

You've already added an object to a bucket and downloaded the object. Now, you create a folder and copy the object and paste it into the folder.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

To copy an object to a folder

1. In the **Buckets** list, choose your bucket name.
2. Choose **Create folder** and configure a new folder:
   a. Enter a folder name (for example, `favorite-pics`).
   b. For the folder encryption setting, choose **Disable**.
   c. Choose **Save**.
3. Navigate to the Amazon S3 bucket or folder that contains the objects that you want to copy.
4. Select the check box to the left of the names of the objects that you want to copy.
5. Choose **Actions** and choose **Copy** from the list of options that appears.

   Alternatively, choose **Copy** from the options that appear.
6. Choose the destination folder:
   a. Choose **Browse S3**.
   b. Choose the option button to the left of the folder name.

      To navigate into a folder and choose a subfolder as your destination, choose the folder name.
   c. Choose **Choose destination**.

   The path to your destination folder appears in the **Destination** box. In **Destination**, you can alternately enter your destination path, for example, s3://*bucket-name*/*folder-name*/.
7. Choose **Copy**.

   Amazon S3 copies your objects to the destination folder.

Next step

To delete an object and a bucket in Amazon S3, see Step 5: Delete your objects and bucket.

Step 5: Delete your objects and bucket

When you no longer need an object or a bucket, we recommend that you delete them to prevent further charges. If you completed this getting started walkthrough as a learning exercise, and you don't plan to use your bucket or objects, we recommend that you delete your bucket and objects so that charges no longer accrue.

Before you delete your bucket, empty the bucket or delete the objects in the bucket. After you delete your objects and bucket, they are no longer available.

If you want to continue to use the same bucket name, we recommend that you delete the objects or empty the bucket, but don't delete the bucket. After you delete a bucket, the name becomes available to reuse. However, another AWS account might create a bucket with the same name before you have a chance to reuse it.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Topics

- Deleting an object
- Emptying your bucket
- Deleting your bucket

Deleting an object

If you want to choose which objects you delete without emptying all the objects from your bucket, you can delete an object.

1. In the **Buckets** list, choose the name of the bucket that you want to delete an object from.
2. Select the object that you want to delete.
3. Choose **Delete** from the options that appear.

4. On the **Delete objects** page, type **`delete`** to confirm deletion of your objects.
5. Choose **Delete objects**.

Emptying your bucket

If you plan to delete your bucket, you must first empty your bucket, which deletes all the objects in the bucket.

To empty a bucket

1. In the **Buckets** list, select the bucket that you want to empty, and then choose **Empty**.
2. To confirm that you want to empty the bucket and delete all the objects in it, in **Empty bucket**, type **`permanently delete`**.

   > **Important**
   >
   > Emptying the bucket cannot be undone. Objects added to the bucket while the empty bucket action is in progress will be deleted.

3. To empty the bucket and delete all the objects in it, and choose **Empty**.

   An **Empty bucket: Status** page opens that you can use to review a summary of failed and successful object deletions.
4. To return to your bucket list, choose **Exit**.

Deleting your bucket

After you empty your bucket or delete all the objects from your bucket, you can delete your bucket.

1. To delete a bucket, in the **Buckets** list, select the bucket.
2. Choose **Delete**.
3. To confirm deletion, in **Delete bucket**, type the name of the bucket.

> **Important**
>
> Deleting a bucket cannot be undone. Bucket names are unique. If you delete your bucket, another AWS user can use the name. If you want to continue to use the same bucket name, don't delete your bucket. Instead, empty and keep the bucket.

4. To delete your bucket, choose **Delete bucket**.

Next steps

In the preceding examples, you learned how to perform some basic Amazon S3 tasks.

The following topics explain the learning paths that you can use to gain a deeper understanding of Amazon S3 so that you can implement it in your applications.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Topics

- Understand common use cases
- Control access to your buckets and objects
- Protect and monitor your storage
- Develop with Amazon S3
- Learn from tutorials
- Explore training and support

Understand common use cases

You can use Amazon S3 to support your specific use case. The AWS Solutions Library and AWS Blog provide use-case specific information and tutorials. The following are some common use cases for Amazon S3:

- **Backup and storage** – Use Amazon S3 storage management features to manage costs, meet regulatory requirements, reduce latency, and save multiple distinct copies of your data for compliance requirements.
- **Application hosting** – Deploy, install, and manage web applications that are reliable, highly scalable, and low-cost. For example, you can configure your Amazon S3 bucket to host a static website. For more information, see Hosting a static website using Amazon S3.
- **Media hosting** – Build a highly available infrastructure that hosts video, photo, or music uploads and downloads.
- **Software delivery** – Host your software applications for customers to download.

Control access to your buckets and objects

Amazon S3 provides a variety of security features and tools. For an overview, see Access control in Amazon S3.

By default, S3 buckets and the objects in them are private. You have access only to the S3 resources that you create. You can use the following features to grant granular resource permissions that support your specific use case or to audit the permissions of your Amazon S3 resources.

- S3 Block Public Access – Block public access to S3 buckets and objects. By default, Block Public Access settings are turned on at the bucket level.
- AWS Identity and Access Management (IAM) identities – Use IAM or AWS IAM Identity Center to create IAM identities in your AWS account to manage access to your Amazon S3 resources. For example, you can use IAM with Amazon S3 to control the type of access that a user or group of users has to an Amazon S3 bucket that your AWS account owns. For more information about IAM identities and best practices, see IAM identities (users, user groups, and roles) in the *IAM User Guide*.
- Bucket policies – Use IAM-based policy language to configure resource-based permissions for your S3 buckets and the objects in them.
- Access control lists (ACLs) – Grant read and write permissions for individual buckets and objects to authorized users. As a general rule, we recommend using S3 resource-based policies (bucket policies and access point policies) or IAM user policies for access control instead of ACLs. Policies are a simplified and more flexible access-control option. With bucket policies and access point policies, you can define rules that apply broadly across all requests to your Amazon S3 resources. For more information about the specific cases when you'd use ACLs instead of resource-based policies or IAM user policies, see Identity and Access Management for Amazon S3.

- S3 Object Ownership – Take ownership of every object in your bucket, simplifying access management for data stored in Amazon S3. S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to disable or enable ACLs. By default, ACLs are disabled. With ACLs disabled, the bucket owner owns all the objects in the bucket and manages access to data exclusively by using access-management policies.
- IAM Access Analyzer for S3 – Evaluate and monitor your S3 bucket access policies, ensuring that the policies provide only the intended access to your S3 resources.

Protect and monitor your storage

- Protecting your storage – After you create buckets and upload objects in Amazon S3, you can protect your object storage. For example, you can use S3 Versioning, S3 Replication, and Multi-Region Access Point failover controls for disaster recovery, AWS Backup to back up your data, and S3 Object Lock to set retention periods, prevent deletions and overwrites, and meet compliance requirements.
- Monitoring your storage – Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon S3 and your AWS solutions. You can monitor storage activity and costs. Also, we recommend that you collect monitoring data from all the parts of your AWS solution so that you can more easily debug a multipoint failure if one occurs.

  You can also use analytics and insights in Amazon S3 to understand, analyze, and optimize your storage usage. For example, use Amazon S3 Storage Lens to understand, analyze, and optimize your storage. S3 Storage Lens provides 29+ usage and activity metrics and interactive dashboards to aggregate data for your entire organization, specific accounts, Regions, buckets, or prefixes. Use Storage Class Analysis to analyze storage access patterns to decide when it's time to move your data to a more cost-effective storage class. To manage your costs, you can use S3 Lifecycle.

Develop with Amazon S3

Amazon S3 is a REST service. You can send requests to Amazon S3 using the REST API or the AWS SDK libraries, which wrap the underlying Amazon S3 REST API, simplifying your programming tasks. You can also use the AWS Command Line Interface (AWS CLI) to make Amazon S3 API calls. For more information, see Making requests in the *Amazon S3 API Reference*.

The Amazon S3 REST API is an HTTP interface to Amazon S3. With the REST API, you use standard HTTP requests to create, fetch, and delete buckets and objects. To use the REST API, you can use any toolkit that supports HTTP. You can even use a browser to fetch objects, as long as they are anonymously readable. For more information, see Developing with Amazon S3 in the *Amazon S3 API Reference*.

To help you build applications using the language of your choice, we provide the following resources.

AWS CLI

You can access the features of Amazon S3 using the AWS CLI. To download and configure the AWS CLI, see Developing with Amazon S3 using the AWS CLI in the *Amazon S3 API Reference*.

The AWS CLI provides two tiers of commands for accessing Amazon S3: High-level (s3) commands and API-level (s3api and `s3control` commands. The high-level S3 commands simplify performing common tasks, such as creating, manipulating, and deleting objects and buckets. The s3api and s3control commands expose direct access to all Amazon S3 API operations, which you can use to carry out advanced operations that might not be possible with the high-level commands alone.

For a list of Amazon S3 AWS CLI commands, see s3, s3api, and s3control.

AWS SDKs and Explorers

You can use the AWS SDKs when developing applications with Amazon S3. The AWS SDKs simplify your programming tasks by wrapping the underlying REST API. The AWS Mobile SDKs and the Amplify JavaScript library are also available for building connected mobile and web applications using AWS.

In addition to the AWS SDKs, AWS Explorers are available for Visual Studio and Eclipse for Java IDE. In this case, the SDKs and the explorers are bundled together as AWS Toolkits.

For more information, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Sample Code and Libraries

The AWS Developer Center and AWS Code Sample Catalog have sample code and libraries written especially for Amazon S3. You can use these code samples to understand how to implement the Amazon S3 API. You can also view the *Amazon Simple Storage Service API Reference* to understand the Amazon S3 API operations in detail.

Learn from tutorials

You can get started with step-by-step tutorials to learn more about Amazon S3. These tutorials are intended for a lab-type environment, and they use fictitious company names, user names, and so on. Their purpose is to provide general guidance. They are not intended for direct use in a production environment without careful review and adaptation to meet the unique needs of your organization's environment.

Getting started

- Tutorial: Storing and retrieving a file with Amazon S3
- Tutorial: Getting started using S3 Intelligent-Tiering
- Tutorial: Getting started using the S3 Glacier storage classes

Optimizing storage costs

- Tutorial: Getting started using S3 Intelligent-Tiering
- Tutorial: Getting started using the S3 Glacier; storage classes
- Tutorial: Optimizing costs and gaining visibility into usage with S3 Storage Lens

Managing storage

- Tutorial: Getting started with Amazon S3 Multi-Region Access Points
- Tutorial: Replicating existing objects in your Amazon S3 buckets with S3 Batch Replication

Hosting videos and websites

- Tutorial: Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53
- Tutorial: Configuring a static website on Amazon S3
- Tutorial: Configuring a static website using a custom domain registered with Route 53

Processing data

- Tutorial: Transforming data for your application with S3 Object Lambda

- Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved
- Tutorial: Batch-transcoding videos with S3 Batch Operations

Protecting data

- Tutorial: Checking the integrity of data in Amazon S3 with additional checksums
- Tutorial: Replicating data within and between AWS Regions using S3 Replication
- Tutorial: Protecting data on Amazon S3 against accidental deletion or application bugs using S3 Versioning, S3 Object Lock, and S3 Replication
- Tutorial: Replicating existing objects in your Amazon S3 buckets with S3 Batch Replication

Explore training and support

You can learn from AWS experts to advance your skills and get expert assistance achieving your objectives.

- **Training** – Training resources provide a hands-on approach to learning Amazon S3. For more information, see AWS training and certification and AWS online tech talks.
- **Discussion Forums** – On the forum, you can review posts to understand what you can and can't do with Amazon S3. You can also post your questions. For more information, see Discussion Forums.
- **Technical Support** – If you have further questions, you can contact Technical Support.

Getting started with Amazon S3 using the AWS CLI

You can get started with Amazon S3 by using the AWS Command Line Interface (AWS CLI) to work with general purpose buckets and objects. A bucket is a container for objects. An object is a file and any metadata that describes that file.

To store an object in Amazon S3, you create a bucket and then upload the object to the bucket. When the object is in the bucket, you can open it, download it, and move it. When you no longer need an object or a bucket, you can clean up your resources.

With Amazon S3, you pay only for what you use. For more information about Amazon S3 features and pricing, see Amazon S3. If you are a new Amazon S3 customer, you can get started with Amazon S3 for free. For more information, see AWS Free Tier.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see Tutorial: Getting started with S3 Express One Zone and Working with directory buckets.

Setting up

Before you begin using the AWS CLI with Amazon S3, make sure that you have:

- Signed up for an AWS account. For instructions, see Setting up Amazon S3.
- Created a user with `s3:*` permissions. For instructions, see Setting up Amazon S3.
- Installed and configured the AWS CLI. For instructions, see Installing or updating the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

To verify that the AWS CLI is properly configured, run the following command:

```
aws sts get-caller-identity
```

For more information, see get-caller-identity in the *AWS CLI Command Reference*.

This command displays a list of available Amazon S3 commands if the AWS CLI is properly installed and configured.

Step 1: Create your first Amazon S3 bucket

After you set up the AWS CLI, you're ready to create a bucket in Amazon S3. Every object in Amazon S3 is stored in a bucket. Before you can store data in Amazon S3, you must create a bucket.

> **Note**
>
> You are not charged for creating a bucket. You are charged only for storing objects in the bucket and for transferring objects in and out of the bucket. The charges that you

incur through following the examples in this guide are minimal (less than $1). For more information about storage charges, see Amazon S3 pricing.

To create a bucket

1. Create a bucket using the `s3api create-bucket` command. Replace *amzn-s3-demo-bucket* with a unique bucket name and `us-east-1` with your desired Region:

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --region us-east-1
```

For Regions other than us-east-1, you need to specify the location constraint:

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --region us-west-2 --create-bucket-configuration LocationConstraint=us-west-2
```

Note

- After you create a bucket, you can't change its Region.
- To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.
- The bucket name must:
  - Be unique within a partition. A partition is a grouping of Regions. AWS currently has three partitions: `aws` (commercial Regions), `aws-cn` (China Regions), and `aws-us-gov` (AWS GovCloud (US) Regions).
  - Be between 3 and 63 characters long.
  - Consist only of lowercase letters, numbers, periods (.), and hyphens (-). For best compatibility, we recommend that you avoid using periods (.) in bucket names, except for buckets that are used only for static website hosting.
  - Begin and end with a letter or number.
- After you create the bucket, you can't change its name.

- Don't include sensitive information in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

2. Verify that your bucket was created by listing all buckets:

```
aws s3 ls
```

3. For Object Ownership, you can disable or enable ACLs and control ownership of objects uploaded to your bucket.

ACLs disabled

To set **Bucket owner enforced** (default) – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the general purpose bucket:

```
aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-
controls="Rules=[{ObjectOwnership=BucketOwnerEnforced}]"
```

> **Note**
>
> By default, ACLs are disabled. A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in circumstances where you must control access for each object individually. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

ACLs enabled

- To set **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL:

```
aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-
controls="Rules=[{ObjectOwnership=BucketOwnerPreferred}]"
```

If you apply the **Bucket owner preferred** setting and want to require all Amazon S3 uploads to include the `bucket-owner-full-control` canned ACL, you can Requiring the bucket-

owner-full-control canned ACL for Amazon S3 PUT operations (bucket owner preferred) that allows only object uploads that use this ACL.

- To set **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs:

```
aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-
controls="Rules=[{ObjectOwnership=ObjectWriter}]"
```

> **Note**
>
> The default setting is **Bucket owner enforced**. To apply the default setting and keep ACLs disabled, only the `s3:CreateBucket` permission is needed. To enable ACLs, you must have the `s3:PutBucketOwnershipControls` permission.

To check the current Object Ownership setting for your bucket:

```
aws s3api get-bucket-ownership-controls --bucket amzn-s3-demo-bucket
```

4. To verify that **Block Public Access** is enabled (it's enabled by default for new buckets):

```
aws s3api get-public-access-block --bucket amzn-s3-demo-bucket
```

By default, all four **Block Public Access** settings are enabled for new buckets. We recommend that you keep all settings enabled, unless you know that you need to turn off one or more of them for your specific use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

If you need to enable **Block Public Access**, use the following command:

```
aws s3api put-public-access-block --bucket amzn-s3-demo-bucket --public-access-
block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=t
```

> **Note**
>
> To enable all Block Public Access settings, only the `s3:CreateBucket` permission is required. To turn off any Block Public Access settings, you must have the `s3:PutBucketPublicAccessBlock` permission.

5. To enable versioning for your bucket:

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-
configuration Status=Enabled
```

By default, Bucket Versioning is disabled. Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your bucket. With versioning, you can recover more easily from both unintended user actions and application failures. For more information about versioning, see Retaining multiple versions of objects with S3 Versioning.

6. Amazon S3 Object Lock helps protect new objects from being deleted or overwritten. For more information, see Locking objects with Object Lock. To enable Locking objects with Object Lock (requires bucket versioning):

For a new bucket:

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket --region us-east-1 --object-
lock-enabled-for-bucket
```

For an existing bucket:

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket --object-lock-
configuration '{"ObjectLockEnabled": "Enabled"}'
```

If you want to set a default Locking objects with Object Lock along with enabling Object Lock, you can use:

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket --object-
lock-configuration '{"ObjectLockEnabled":"Enabled","Rule":{"DefaultRetention":
{"Mode":"COMPLIANCE","Days":30}}}'
```

You can replace "COMPLIANCE" with "GOVERNANCE" for a less restrictive mode, and adjust the number of days as needed.

> **Note**
>
> To create an Object Lock enabled bucket, you must have the following permissions: `s3:CreateBucket`, `s3:PutBucketVersioning`, and `s3:PutBucketObjectLockConfiguration`.

7. You can add tags to your bucket. With AWS cost allocation, you can use bucket tags to annotate billing for your use of a bucket. A tag is a key-value pair that represents a label that you assign to a bucket. For more information, see Using cost allocation S3 bucket tags.

   To add tags to your bucket:

   ```
   aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging
    'TagSet=[{Key=Purpose,Value=Testing},{Key=Environment,Value=Development}]'
   ```

8. Buckets and new objects are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption configuration. To verify the default encryption of your bucket, use the following command:

   ```
   aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket
   ```

   You can also configure server-side encryption with AWS KMS keys (SSE-KMS) and dual-layer server-side encryption with AWS KMS keys (DSSE-KMS) for your bucket. Both the AWS managed key (aws/s3) and your customer managed keys can be used as your AWS KMS key for SSE-KMS and DSSE-KMS encryption configuration. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*. For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

   > **Important**
   >
   > The AWS KMS key must be in the same AWS Region as your Amazon S3 bucket. Cross-region KMS keys aren't supported for Amazon S3 bucket encryption.

When you configure your bucket to use default encryption with SSE-KMS, you can also use Amazon S3 Bucket Keys. Amazon S3 Bucket Keys lower the cost of encryption by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys. Amazon S3 Bucket Keys aren't supported for DSSE-KMS. In the AWS CLI, Amazon S3 Bucket Keys are NOT enabled by default when creating a new bucket. This is different from the console behavior where they are enabled by default.

To configure SSE-KMS and enable Amazon S3 Bucket Keys:

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-
encryption-configuration "{\"Rules\":[{\"ApplyServerSideEncryptionByDefault
\":{\"SSEAlgorithm\":\"aws:kms\",\"KMSMasterKeyID\":\"YOUR-KMS-KEY-ARN\"},
\"BucketKeyEnabled\":true}]}"
```

To check whether Amazon S3 Bucket Keys are enabled for a bucket:

```
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket
```

The output will include a `BucketKeyEnabled` field set to either `true` or `false`.

To configure DSSE-KMS, use the following command:

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-
encryption-configuration '{"Rules":[{"ApplyServerSideEncryptionByDefault":
{"SSEAlgorithm":"aws:kms:dsse","KMSMasterKeyID":"YOUR-KMS-KEY-ARN"}}]}'
```

For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets. For more information about SSE-S3, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

> **Important**
>
> If you use the SSE-KMS or DSSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quota of AWS KMS. You can reduce KMS API calls by enabling Amazon S3 Bucket Keys, which decreases the number of requests sent to AWS KMS. For more information about AWS KMS quotas and how to

request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.

You've created a bucket in Amazon S3. Next step is to upload an object to your bucket.

Step 2: Upload an object to your bucket

After creating a bucket in Amazon S3, you're ready to upload an object to the bucket. An object can be any kind of file: a text file, a photo, a video, and so on.

To upload an object to a bucket

1. Create a simple text file to upload. You can use any text editor or run the following command:

```
echo 'Hello, Amazon S3!' > example.txt
```

2. Upload the file to your bucket using the s3 cp command:

```
aws s3 cp example.txt s3://amzn-s3-demo-bucket/
```

   If the upload is successful, you'll see output similar to:

```
upload: ./example.txt to s3://amzn-s3-demo-bucket/example.txt
```

3. Verify that the object was uploaded by listing the contents of your bucket:

```
aws s3 ls s3://amzn-s3-demo-bucket/
```

You've successfully uploaded an object to your bucket. Next step is to download an object.

Step 3: Download an object

After you upload an object to a bucket, you can view information about your object and download the object to your local computer.

To download an object from an Amazon S3 bucket

1. To get information about your object:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key example.txt
```

This command returns metadata about the object, including its content type, content length, and last modified date.

2. Download the object to your local computer:

```
aws s3 cp s3://amzn-s3-demo-bucket/example.txt downloaded-example.txt
```

If the download is successful, you'll see output similar to:

```
download: s3://amzn-s3-demo-bucket/example.txt to ./downloaded-example.txt
```

3. Verify the contents of the downloaded file:

```
cat downloaded-example.txt
```

Note

- Unlike the console, the AWS CLI can download multiple objects at once using wildcards or the `--recursive` flag.
- When downloading objects with the AWS CLI, periods (.) at the end of object key names are preserved, unlike in the console where they are removed. This is important if your object keys end with periods.

Example of downloading multiple objects:

To download multiple objects from an Amazon S3 bucket with specific file extensions, use the recursive copy command with `exclude` and `include` filters as shown in the example.

```
aws s3 cp s3://amzn-s3-demo-bucket/ . --recursive --exclude "*" --include "*.txt"
```

You've successfully downloaded your object. Next step is to copy your object to a folder.

Step 4: Copy your object to a folder

You've already added an object to a bucket and downloaded the object. Now, you create a folder and copy the object to the folder.

To copy an object to a folder

1. In Amazon S3, folders are represented by prefixes in object keys. Create a "folder" by copying an object with a prefix:

   ```
   aws s3 cp s3://amzn-s3-demo-source-bucket/example.txt s3://amzn-s3-demo-destination-bucket/favorite-files/example.txt
   ```

   If the copy is successful, you'll see output similar to:

   ```
   copy: s3://amzn-s3-demo-source-bucket/example.txt to s3://amzn-s3-demo-destination-bucket/favorite-files/example.txt
   ```

2. Verify that the object was copied by listing the contents of the folder:

   ```
   aws s3 ls s3://amzn-s3-demo-destination-bucket/favorite-files/
   ```

You've successfully copied your object to a folder. Next step is to delete your objects and bucket.

Step 5: Delete your objects and bucket

When you no longer need an object or a bucket, we recommend that you delete them to prevent further charges. If you completed this getting started walkthrough as a learning exercise, and you don't plan to use your bucket or objects, we recommend that you delete your bucket and objects so that charges no longer accrue.

Before you delete your bucket, empty the bucket or delete the objects in the bucket. After you delete your objects and bucket, they are no longer available.

If you want to continue to use the same bucket name, we recommend that you delete the objects or empty the bucket, but don't delete the bucket. After you delete a bucket, the name becomes available to reuse. However, another AWS account might create a bucket with the same name before you have a chance to reuse it.

Deleting an object

If you want to choose which objects you delete without emptying all the objects from your bucket, you can delete an object.

Delete a specific object:

```
aws s3 rm s3://amzn-s3-demo-bucket/example.txt
```

If the deletion is successful, you'll see output similar to:

```
delete: s3://amzn-s3-demo-bucket/example.txt
```

Emptying your bucket

If you plan to delete your bucket, you must first empty your bucket, which deletes all the objects, versions, and delete markers in the bucket.

To empty a bucket

> **Important**
>
> Emptying the bucket cannot be undone. Objects added to the bucket while the empty bucket action is in progress will be deleted.

1. **Option 1:** For smaller buckets, use the `rm` command with the `--recursive` flag to delete all objects in the bucket:

   ```
   aws s3 rm s3://amzn-s3-demo-bucket --recursive
   ```

   This command deletes all objects in the bucket, including objects in folders.

   > **Note**
   >
   > If your bucket contains many objects or large objects, this command might time out. For buckets with large amounts of data, use the Amazon S3 Lifecycle rule to expire objects in the buckets.

Option 2: Use Amazon S3 Lifecycle rules (recommended for large buckets)

For buckets with many objects or large objects, use an Amazon S3 Lifecycle rule to automatically expire and delete all objects. Wait for the lifecycle rule to process (this may take up to 24 hours). For more information about using lifecycle rules to empty buckets, see How do I empty an Amazon S3 bucket using a lifecycle configuration rule?.

2. Verify that the bucket is empty:

```
aws s3 ls s3://amzn-s3-demo-bucket
```

3. If your bucket has versioning enabled, use the following commands to delete versioned objects and delete markers.

   Remove versioned objects:

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete "$(aws s3api list-
object-versions --bucket amzn-s3-demo-bucket --output json --query='{Objects:
 Versions[].{Key:Key,VersionId:VersionId}}')"
```

   Remove delete markers:

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete "$(aws s3api list-
object-versions --bucket amzn-s3-demo-bucket --output json --query='{Objects:
 DeleteMarkers[].{Key:Key,VersionId:VersionId}}')"
```

4. Verify that the bucket is empty of all object versions and delete markers:

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket
```

   The output should show no versions or delete markers remaining.

Deleting your bucket

After you empty your bucket or delete all the objects from your bucket, you can delete your bucket.

> **Important**
>
> Deleting a bucket can't be undone. Bucket names are unique. If you delete your bucket, another AWS user can use the name. If you want to continue to use the same bucket name, don't delete your bucket. Instead, empty and keep the bucket.

To delete your bucket

1. Delete your bucket:

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket
```

2. Verify that the bucket was deleted by listing all your buckets:

```
aws s3 ls
```

Next steps

In the preceding examples, you learned how to perform some basic Amazon S3 tasks using the AWS CLI.

The following topics explain the learning paths that you can use to gain a deeper understanding of Amazon S3 so that you can implement it in your applications.

- Understand common use cases
- Control access to your buckets and objects
- Protect and monitor your storage
- Develop with Amazon S3
- Understand common use cases
- Explore training and support

The following list shows common AWS CLI commands for Amazon S3:

- cp – Copies files or objects between your local file system and Amazon S3, or between Amazon S3 locations

- ls – Lists Amazon S3 objects and common prefixes under a specified bucket and prefix
- mb – Creates an Amazon S3 bucket
- mv – Moves files or objects between your local file system and Amazon S3, or between Amazon S3 locations
- presign – Generates a pre-signed URL for an Amazon S3 object that allows temporary access without AWS credentials
- rb – Removes an empty Amazon S3 bucket. You can use the `--force` flag to automatically empty and delete a bucket with contents in a single command. This action can't be undone.
- rm – Deletes objects from Amazon S3
- sync – Syncs directories and Amazon S3 prefixes by recursively copying new and updated files from the source directory to the destination.
- website – Configures a bucket as a static website

For more information about the AWS CLI commands for Amazon S3, see the following resources:

- s3 – High-level Amazon S3 commands that simplify common operations
- s3api – Direct access to all Amazon S3 API operations
- s3control – Direct access to all Amazon S3 Control API operations

Creating, configuring, and working with Amazon S3 general purpose buckets

To store your data in Amazon S3, you work with resources known as buckets and objects. A *bucket* is a container for objects. An *object* is a file and any metadata that describes that file.

To store an object in Amazon S3, you create a bucket and then upload the object to a bucket. When the object is in the bucket, you can open it, download it, and move it. When you no longer need an object or a bucket, you can clean up your resources.

The topics in this section provide an overview of working with general purpose buckets in Amazon S3. They include information about naming, creating, accessing, and deleting general purpose buckets. For more information about viewing or listing objects in a bucket, see Organizing, listing, and working with your objects.

There are several types of Amazon S3 buckets. Before creating a bucket, make sure that you choose the bucket type that best fits your application and performance requirements. For more information about the various bucket types and the appropriate use cases for each, see Buckets.

Note

For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Note

With Amazon S3, you pay only for what you use. For more information about Amazon S3 features and pricing, see Amazon S3. If you are a new Amazon S3 customer, you can get started with Amazon S3 for free. For more information, see AWS Free Tier.

Topics

- General purpose buckets overview
- Namespaces for general purpose buckets
- Common general purpose bucket patterns for building applications on Amazon S3

- General purpose bucket naming rules
- General purpose bucket quotas, limitations, and restrictions
- Accessing an Amazon S3 general purpose bucket
- Creating a general purpose bucket
- Viewing the properties for an S3 general purpose bucket
- Listing Amazon S3 general purpose buckets
- Emptying a general purpose bucket
- Deleting a general purpose bucket
- Mount an Amazon S3 bucket as a local file system
- Working with Storage Browser for Amazon S3
- Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration
- Using Requester Pays general purpose buckets for storage transfers and usage

General purpose buckets overview

To upload your data (photos, videos, documents, etc.) to Amazon S3, you must first create an S3 bucket in one of the AWS Regions.

There are several types of Amazon S3 buckets. Before creating a bucket, make sure that you choose the bucket type that best fits your application and performance requirements. For more information about the various bucket types and the appropriate use cases for each, see Buckets.

The following sections provide more information about general purpose buckets, including bucket naming rules, quotas, and bucket configuration details. For a list of restriction and limitations related to Amazon S3 buckets see, General purpose bucket quotas, limitations, and restrictions.

Topics

- General purpose buckets overview
- Common general purpose bucket patterns
- Permissions
- Managing public access to general purpose buckets
- Managing tags for general purpose buckets
- General purpose buckets configuration options

- General purpose buckets operations
- General purpose buckets performance monitoring

General purpose buckets overview

Every object is contained in a bucket. For example, if the object named `photos/puppy.jpg` is stored in the `amzn-s3-demo-bucket` general purpose bucket in the US West (Oregon) Region, then it is addressable by using the URL `https://amzn-s3-demo-bucket.s3.us-west-2.amazonaws.com/photos/puppy.jpg`. For more information, see Accessing a Bucket.

- General purpose bucket quotas for commercial Regions can only be viewed and managed from US East (N. Virginia).
- General purpose bucket quotas for AWS GovCloud (US) can only be viewed and managed from AWS GovCloud (US-West).

In terms of implementation, buckets and objects are AWS resources, and Amazon S3 provides APIs for you to manage them. For example, you can create a bucket and upload objects using the Amazon S3 API. You can also use the Amazon S3 console to perform these operations. The console uses the Amazon S3 APIs to send requests to Amazon S3.

This section describes how to work with general purpose buckets. For information about working with objects, see Amazon S3 objects overview.

By default, general purpose buckets exist in a global namespace, which means that each bucket name must be unique across all AWS accounts in all the AWS Regions within a partition. A partition is a grouping of Regions. AWS currently has four partitions: `aws` (Standard Regions), `aws-cn` (China Regions), `aws-us-gov` (AWS GovCloud (US)), and `aws-eusc` (European Sovereign Cloud). After creating a general purpose bucket in the shared global namespace, that bucket name is unavailable for anyone else to create within partition. When a bucket owner deletes their bucket, the bucket name becomes available again in the global namespace for anyone to re-create.

Alternatively, you can create buckets in your reserved account regional namespace to easily create predictable bucket names with assurance that the names you want will always be available for you to use. Your account regional namespace is a subdivision of the global namespace that only your account can use. By creating new buckets in your account regional namespace, you have assurance that your desired bucket names will always be available for you to use. For more information on account regional namespaces, see Namespaces for general purpose buckets.

After a general purpose bucket is created, the name of that bucket cannot be used by another AWS account in the same partition until the bucket is deleted. You should not depend on specific bucket naming conventions for availability or security verification purposes. For bucket naming guidelines, see General purpose bucket naming rules.

Amazon S3 creates buckets in a Region that you specify. To reduce latency, minimize costs, or address regulatory requirements, choose any AWS Region that is geographically close to you. For example, if you reside in Europe, you might find it advantageous to create buckets in the Europe (Ireland) or Europe (Frankfurt) Regions. For a list of Amazon S3 Regions, see Regions and Endpoints in the *AWS General Reference*.

> **Note**
>
> Objects that belong to a bucket that you create in a specific AWS Region never leave that Region, unless you explicitly transfer them to another Region. For example, objects that are stored in the Europe (Ireland) Region never leave it.

Common general purpose bucket patterns

When you build applications on Amazon S3, you can use unique general purpose buckets to separate different datasets or workloads. Depending on your use case, there are different design patterns and best practices for using general purpose buckets. For more information, see Common general purpose bucket patterns for building applications on Amazon S3.

Permissions

You can use your AWS account root user credentials to create a general purpose bucket and perform any other Amazon S3 operation. However, we recommend that you do not use the root user credentials of your AWS account to make requests, such as to create a bucket. Instead, create an AWS Identity and Access Management (IAM) user, and grant that user full access (users by default have no permissions).

These users are referred to as *administrators*. You can use the administrator user credentials, instead of the root user credentials of your account, to interact with AWS and perform tasks, such as create a bucket, create users, and grant them permissions.

For more information, see AWS account root user credentials and IAM user credentials in the *AWS General Reference* and Security best practices in IAM in the *IAM User Guide*.

The AWS account that creates a resource owns that resource. For example, if you create an IAM user in your AWS account and grant the user permission to create a bucket, the user can create a bucket. But the user does not own the bucket; the AWS account that the user belongs to owns the bucket. The user needs additional permission from the resource owner to perform any other bucket operations. For more information about managing permissions for your Amazon S3 resources, see Identity and Access Management for Amazon S3.

Managing public access to general purpose buckets

Public access is granted to general purpose buckets and objects through bucket policies, access control lists (ACLs), or both. To help you manage public access to Amazon S3 resources, Amazon S3 provides settings to block public access. Amazon S3 Block Public Access settings can override ACLs and bucket policies so that you can enforce uniform limits on public access to these resources. You can apply Block Public Access settings to individual buckets or to all buckets in your account.

To ensure that all of your Amazon S3 general purpose buckets and objects have their public access blocked, all four settings for Block Public Access are enabled by default when you create a new bucket. We recommend that you turn on all four settings for Block Public Access for your account too. These settings block all public access for all current and future buckets.

Before applying these settings, verify that your applications will work correctly without public access. If you require some level of public access to your buckets or objects—for example, to host a static website, as described at Hosting a static website using Amazon S3—you can customize the individual settings to suit your storage use cases. For more information, see Blocking public access to your Amazon S3 storage.

However, we highly recommend keeping Block Public Access enabled. If you want to keep all four Block Public Access settings enabled and host a static website, you can use Amazon CloudFront origin access control (OAC). Amazon CloudFront provides the capabilities required to set up a secure static website. Amazon S3 static websites support only HTTP endpoints. Amazon CloudFront uses the durable storage of Amazon S3 while providing additional security headers, such as HTTPS. HTTPS adds security by encrypting a normal HTTP request and protecting against common cyberattacks.

For more information, see Getting started with a secure static website in the *Amazon CloudFront Developer Guide*.

Note

If you see an `Error` when you list your general purpose buckets and their public access settings, you might not have the required permissions. Make sure that you have the following permissions added to your user or role policy:

```
s3:GetAccountPublicAccessBlock
s3:GetBucketPublicAccessBlock
s3:GetBucketPolicyStatus
s3:GetBucketLocation
s3:GetBucketAcl
s3:ListAccessPoints
s3:ListAllMyBuckets
```

In some rare cases, requests can also fail because of an AWS Region outage.

Managing tags for general purpose buckets

You can add tags to your Amazon S3 buckets to categorize and track your AWS costs or for access control. You can use tags as cost allocation tags to track storage costs in AWS Billing and Cost Management. You can also use tags for attribute-based access control (ABAC), to scale access permissions and grant access to S3 buckets based on their tags.

For more information, see Using tags with S3 general purpose buckets

General purpose buckets configuration options

Amazon S3 supports various options for you to configure your general purpose bucket. For example, you can configure your bucket for website hosting, add a configuration to manage the lifecycle of objects in the bucket, and configure the bucket to log all access to the bucket. Amazon S3 supports subresources for you to store and manage the bucket configuration information. You can use the Amazon S3 API to create and manage these subresources. However, you can also use the console or the AWS SDKs.

Note

There are also object-level configurations. For example, you can configure object-level permissions by configuring an access control list (ACL) specific to that object.

These are referred to as subresources because they exist in the context of a specific bucket or object. The following table lists subresources that enable you to manage bucket-specific configurations.

Subresource	Description
cors (cross-origin resource sharing)	You can configure your bucket to allow cross-origin requests. For more information, see Using cross-origin resource sharing (CORS).
event notification	You can enable your bucket to send you notifications of specified bucket events. For more information, see Amazon S3 Event Notifications.
lifecycle	You can define lifecycle rules for objects in your bucket that have a well-defined lifecycle. For example, you can define a rule to archive objects one year after creation, or delete an object 10 years after creation. For more information, see Managing the lifecycle of objects.
location	When you create a bucket, you specify the AWS Region where you want Amazon S3 to create the bucket. Amazon S3 stores this information in the location subresource and provides an API for you to retrieve this infor mation.
logging	Logging enables you to track requests for access to your bucket. Each access log record provides details about a single access request, such as the requester, bucket name, request time, request action, response status, and error code, if any. Access log information can be useful in security and access audits. It can also help you learn about your customer base and understand your Amazon S3 bill. For more information, see Logging requests with server access logging.
object locking	To use S3 Object Lock, you must enable it for a bucket. You can also optionally configure a default retention mode and period that applies to new objects that are placed in the bucket.

Subresource	Description
	For more information, see Locking objects with Object Lock.
policy and *ACL* (access control list)	All your resources (such as buckets and objects) are private by default . Amazon S3 supports both bucket policy and access control list (ACL) options for you to grant and manage bucket-level permissions. Amazon S3 stores the permission information in the *policy* and *acl* subresources. For more information, see Identity and Access Management for Amazon S3.
replication	Replication is the automatic, asynchronous copying of objects across buckets in different or the same AWS Regions. For more information, see Replicating objects within and across Regions.
requestPayment	By default, the AWS account that creates the bucket (the bucket owner) pays for downloads from the bucket. Using this subresource, the bucket owner can specify that the person requesting the download will be charged for the download. Amazon S3 provides an API for you to manage this subresource. For more information, see Using Requester Pays general purpose buckets for storage transfers and usage.
tagging	You can add tags to your Amazon S3 buckets to categorize and track your AWS costs or for access control. You can use tags as cost allocation tags to track storage costs in AWS Billing and Cost Management. You can also use tags for attribute-based access control (ABAC), to scale access permissions and grant access to S3 buckets based on their tags. For more information, see Using tags with S3 general purpose buckets.
transfer acceleration	Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. Transfer Acceleration takes advantage of the globally distributed edge locations of Amazon CloudFront. For more information, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

Subresource	Description
versioning	Versioning helps you recover accidental overwrites and deletes. We recommend versioning as a best practice to recover objects from being deleted or overwritten by mistake. For more information, see Retaining multiple versions of objects with S3 Versioning.
website	You can configure your bucket for static website hosting. Amazon S3 stores this configuration by creating a *website* subresource. For more information, see Hosting a static website using Amazon S3.

General purpose buckets operations

The high availability engineering of Amazon S3 is focused on *get*, *put*, *list*, and *delete* operations. Because general purpose bucket operations work against a centralized, global resource space, we recommend that you don't create, delete, or configure buckets on the high availability code path of your application. It's better to create, delete, or configure buckets in a separate initialization or setup routine that you run less often.

General purpose buckets performance monitoring

When you have critical applications and business processes that rely on AWS resources, it's important to monitor and get alerts for your system. Monitoring your data can help maintain the reliability, availability, and performance of Amazon S3 and your AWS solutions. There are several AWS services that you can use to collect and aggregates metrics and logs for your S3 buckets.

Depending on your use case, you can choose which AWS service best suits your organization's needs to debug issues, monitor your data, optimize storage costs, or troubleshoot multi-point issues. For example:

- **To improve the performance of applications that use S3:** Set up CloudWatch alarms to monitor your storage data, replication metrics, or request metrics.
- **To plan for storage usage, optimize storage costs, or to find out how much storage you have across your entire organization:** Use Amazon S3 Storage Lens. Alternatively, you can use S3

Storage Lens to improve your data performance by enabling advanced metrics and using the detailed status-code metrics to get counts for successful or failed requests.

- **For a unified view of your operational health:** Publish S3 Storage Lens usage and activity metrics to a Amazon CloudWatch dashboard.

> **Note**
>
> The Amazon CloudWatch publishing option is available for S3 Storage Lens dashboards upgraded to **Advanced metrics and recommendations**. You can enable the CloudWatch publishing option for a new or existing dashboard configuration in S3 Storage Lens.

- **To obtain a record of actions taken by a user, role, or an AWS service:** Set up AWS CloudTrail logs. You can also use AWS CloudTrail logs to review API calls for Amazon S3 as events.
- **To receive notifications about when a certain event happens in your S3 bucket:** Set up Amazon S3 event notifications.
- **To obtain detailed records for the requests that are made to an S3 bucket:** Set up S3 access logs.

For a list of all the different AWS services that you can use to monitor your data, see Logging and monitoring in Amazon S3.

Namespaces for general purpose buckets

By default, general purpose buckets exist in a global namespace. This means that each bucket name must be unique across all AWS accounts in all the AWS Regions within a partition. A partition is a grouping of Regions. AWS currently has four partitions: `aws` (Standard Regions), `aws-cn` (China Regions), `aws-us-gov` (AWS GovCloud (US)), and `aws-eusc` (European Sovereign Cloud). When you create a general purpose bucket, you can choose to create a bucket in the shared global namespace. You can also choose to create a bucket in your account regional namespace. Your account regional namespace is a subdivision of the global namespace that only your account can create buckets in.

Topics

- Global general purpose buckets
- Account regional namespace general purpose buckets
- Restrictions and considerations

- AWS Region Code Format
- Requiring the creation of buckets in your account regional namespace
- Creating a bucket in your account regional namespace

Global general purpose buckets

By default, you create global general purpose buckets in the shared global namespace. After creating a general purpose bucket in the shared global namespace, that bucket name is unavailable for anyone else to create within the partition. When you delete a global general purpose bucket, the bucket name becomes available again in the global namespace for anyone to re-create.

When creating global general purpose buckets, you can request any name that adheres to the bucket naming rules. These rules include specifying a name between 3 (minimum) and 63 (maximum) characters long. The name can only consist of lowercase letters, numbers, periods (.), and hyphens (-). Bucket names must begin and end with a letter or number and cannot contain two adjacent periods. For more information on bucket naming rules, see General purpose bucket naming rules.

When specifying a global general purpose bucket name, you must select a unique name that is not already in use for the partition. If you attempt to create a bucket that already exists and is owned by someone else, you will receive an HTTP 409 BucketAlreadyExists error. If you attempt to create a bucket that already exists and is owned by you, you will receive an HTTP 409 BucketAlreadyOwnedByYou error.

You can create global general purpose buckets to have the most flexibility in selecting your requested bucket names. Since it's a shared global namespace, other accounts can create similar bucket names. Other accounts can also re-create bucket names that you previously deleted. You should not depend on specific bucket naming conventions for availability or security verification purposes. Don't write code assuming your chosen bucket name is available unless you have already created the bucket. One method for creating bucket names that aren't predictable is to append a Globally Unique Identifier (GUID) to your bucket name. For example, `amzn-s3-demo-bucket-a1b2c3d4-5678-90ab-cdef-example11111`. For more information, see Creating a bucket that uses a GUID in the bucket name.

Account regional namespace general purpose buckets

Although Amazon S3 general purpose buckets exist in a shared global namespace, you can optionally create buckets in your account regional namespace. The account regional namespace

is a reserved subdivision of the global bucket namespace. Only your account can create general purpose buckets in this namespace. New general purpose buckets created in your account regional namespace are unique to your account. These buckets can never be re-created by another account. These buckets support all the S3 features and AWS services that general purpose buckets in the shared global namespace already support. Your applications require no change to interact with buckets in your account regional namespace.

> **Note**
>
> You can create buckets in your account regional namespace in all AWS Regions except Middle East (Bahrain) and Middle East (UAE).

Creating buckets in your account regional namespace is a security best practice. These bucket names can only ever be used by your account. You can create buckets in your account regional namespace to easily template general purpose bucket names across multiple AWS Regions. You can have assurance that no other account can create bucket names in your namespace. If another account attempts to create a bucket with your account regional suffix, the CreateBucket request will be rejected.

Account regional namespace naming convention

General purpose buckets in your account regional namespace must follow a specific naming convention. These buckets consist of a bucket name prefix that you create and a suffix that contains your 12-digit AWS Account ID, the AWS Region code, and ends with `-an`.

```
bucket-name-prefix-accountId-region-an
```

For example, the following general purpose bucket exists in the account regional namespace for AWS Account 111122223333 in the us-west-2 Region:

```
amzn-s3-demo-bucket-111122223333-us-west-2-an
```

To create a bucket in your account regional namespace, you make a CreateBucket request. Specify the `x-amz-bucket-namespace` request header with the value set to `account-regional`. Also specify an account regional namespace formatted bucket name: `<customer-chosen-name>-<AWS-Account-ID>-<AWS-Region>-an`.

Note

When you create a general purpose bucket in your account regional namespace using the console, a suffix is automatically added to the bucket name prefix that you provide. This suffix includes your AWS Account ID and the AWS Region that you selected to create your bucket in. When you create a general purpose bucket in your account regional namespace using the CreateBucket API, you must provide the full suffix. This includes your AWS Account ID and the AWS Region in your request. For a list of the AWS Region codes, see AWS Region Code Format.

Integrating the account regional namespace to your CloudFormation templates

You can update your infrastructure-as-code tools, like CloudFormation, to simplify creating buckets in your account regional namespace. CloudFormation offers the pseudo parameters `AWS::AccountId` and `AWS::Region`. These parameters make it easy to build CloudFormation templates that create account regional namespace buckets. For more information, see Get AWS values using pseudo parameters.

Example 1 using BucketName with Sub:

```
BucketName: !Sub "amzn-s3-demo-bucket-${AWS::AccountId}-${AWS::Region}-an"
BucketNamespace: "account-regional"
```

Example 2 using BucketNamePrefix:

```
BucketNamePrefix: 'amzn-s3-demo-bucket'
BucketNamespace: "account-regional"
```

Alternatively, you can also use the BucketNamePrefix property to update your CloudFormation template. The BucketNamePrefix lets you simply provide the customer-defined portion of the bucket name. It then automatically adds the account regional namespace suffix based on the requesting AWS Account and AWS Region specified.

Using these options, you can build a custom CloudFormation template to easily create general purpose buckets in your account regional namespace. For more information, see AWS::S3::Bucket in the CloudFormation User Guide.

Restrictions and considerations

When creating buckets in the shared global namespace, the following considerations apply:

- A bucket name in the shared global namespace can't be used by another AWS account in the same partition until the bucket is deleted. After you delete a bucket in the shared global namespace, be aware that another AWS account in the same partition can use the same bucket name for a new bucket and can therefore potentially receive requests intended for the deleted bucket.
- When building applications that will create buckets in the shared global namespace, make sure to consider that your desired bucket names may be already taken by another account and that other accounts may have bucket names that are similar to yours.
- Because Amazon S3 identifies buckets based on their names, an application that uses an incorrect bucket name in a request could inadvertently perform operations against a different bucket than expected. To help avoid unintentional bucket interactions in situations like this, you can use bucket owner condition. For more information, see Verifying bucket ownership with bucket owner condition.

When creating buckets in your account regional namespace, the following restrictions and considerations apply:

- Any attempt to re-create an account regional namespace bucket that you already own in any AWS Region will return an HTTP 409 BucketAlreadyOwnedByYou error.
- You should use the S3 regional endpoints to create buckets in your account regional namespace. For backwards compatibility, you can use the legacy global endpoint to create buckets in your account regional namespace in the US East (N. Virginia) Region.
- Your account regional suffix counts towards the maximum number of 63-characters allowed in general purpose bucket names. So if your account regional suffix is `-012345678910-us-east-1-an`, then you have 37-characters available for your bucket name prefix.

AWS Region Code Format

To create a bucket in your account regional namespace, you must include the AWS Region in the suffix where you want to create the general purpose bucket. You must specify the full AWS Region code (for example, `us-west-2`) in the suffix. See AWS Regions for an entire list of the AWS Region

codes. The following bucket names show two examples of the AWS Region code format that you must use when creating buckets in your account regional namespace:

- `amzn-s3-demo-bucket-012345678910-ap-southeast-1-an`
- `amzn-s3-demo-bucket-987654321012-eu-north-1-an`

Requiring the creation of buckets in your account regional namespace

You can enforce that your IAM principals only create buckets in your account regional namespace. Use the `s3:x-amz-bucket-namespace` condition key. The following examples show how you can enforce account regional bucket creation in an IAM policy, Resource Control Policy, or Service Control Policy.

IAM policy

The following IAM policy denies the s3:CreateBucket permission to the IAM principal if the request does not include the x-amz-bucket-namespace header set to account-regional.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RequireAccountRegionalBucketCreation",
      "Effect": "Deny",
      "Action": "s3:CreateBucket",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-bucket-namespace": "account-regional"
        }
      }
    }
  ]
}
```

Resource Control Policy

The following Resource Control Policy denies the s3:CreateBucket permission to everyone if the request does not include the x-amz-bucket-namespace header set to account-regional.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "OnlyCreateBucketsInAccountRegionalNamespace",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:CreateBucket",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-bucket-namespace": "account-regional"
                }
            }
        }
    ]
}
```

Service Control Policy

The following Service Control Policy denies the s3:CreateBucket permission to everyone if the request does not include the x-amz-bucket-namespace header set to account-regional.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireAccountRegionalBucketCreation",
            "Effect": "Deny",
            "Action": "s3:CreateBucket",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-bucket-namespace": "account-regional"
                }
            }
        }
    ]
}
```

Creating a bucket in your account regional namespace

The following examples show you how to create a general purpose bucket in your account regional namespace.

Using the AWS CLI

The following AWS CLI example creates a general purpose bucket in the account regional namespace for AWS Account 012345678910 in the US West (N. California) Region (us-west-1). To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket-012345678910-us-west-1-an \
    --bucket-namespace account-regional
    --region us-west-1 \
    --create-bucket-configuration LocationConstraint=us-west-1
```

Common general purpose bucket patterns for building applications on Amazon S3

When you build applications on Amazon S3, you can use unique general purpose buckets to separate different datasets or workloads. When you build applications that serve end users or different user groups, use our best practices design patterns to build applications that can best take advantage of Amazon S3 features and scalability.

> **Important**
>
> We recommend that you create general purpose bucket names that are not predictable. Do not write code assuming your chosen bucket name is available unless you have already created the bucket. We recommend creating buckets in your account regional namespace for assurance that only your account can ever own these bucket names, see Namespaces for general purpose buckets. For more information about general purpose bucket naming rules, see General purpose bucket naming rules.

Topics

- Multi-tenant general purpose bucket pattern
- Bucket-per-use pattern

Multi-tenant general purpose bucket pattern

With multi-tenant buckets, you create a single general purpose bucket for a team or workload. You use unique S3 prefixes to organize the objects that you store in the bucket. A prefix is a string of characters at the beginning of the object key name. A prefix can be any length, subject to the maximum length of the object key name (1,024 bytes). You can think of prefixes as a way to organize your data in a similar way to directories. However, prefixes are not directories.

For example, to store information about cities, you might organize it by continent, then by country, then by province or state. Because these names don't usually contain punctuation, you might use slash (/) as the delimiter. The following examples shows prefixes being used to organize city names by continent, country, and then province or state, using a slash (/) delimiter.

- Europe/France/NouvelleA-Aquitaine/Bordeaux
- North America/Canada/Quebec/Montreal
- North America/USA/Washington/Bellevue
- North America/USA/Washington/Seattle

This pattern scales well when you have hundreds of unique datasets within a general purpose bucket. With prefixes, you can easily organize and group these datasets.

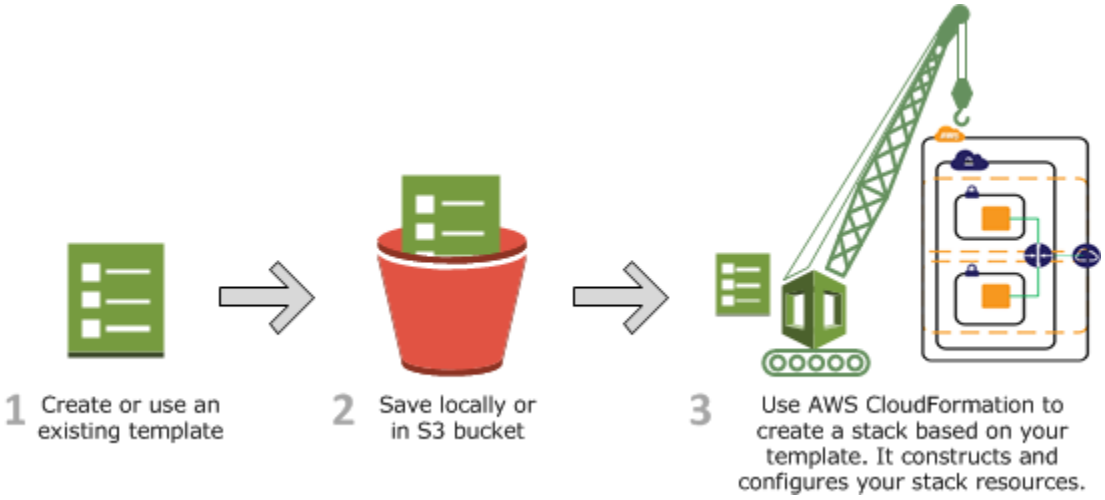
However, one potential drawback to the multi-tenant general purpose bucket pattern is that many S3 bucket-level features like default bucket encryption, S3 Versioning, and S3 Requester Pays are set at the bucket-level and not the prefix-level. If the different datasets within the multi-tenant bucket have unique requirements, the fact that you can't configure many S3 bucket-level features at the prefix-level can make it difficult for you to specify the correct settings for each dataset. Additionally, in a multi-tenant bucket, cost allocation can become complex as you work to understand the storage, requests, and data transfer associated with specific prefixes.

Bucket-per-use pattern

With the bucket-per-use pattern, you create a general purpose bucket for each distinct dataset, end user, or team. Because you can configure S3 bucket-level features for each of these buckets, you can use this pattern to configure unique bucket-level settings. For example, you can configure features like default bucket encryption, S3 Versioning, and S3 Requester Pays in a way that is customized to the dataset in each bucket. Using one bucket for each distinct dataset, end user, or team can also help you simplify both your access management and cost allocation strategies.

A potential drawback to this strategy is that you will need to manage potentially thousands of buckets. All AWS accounts have a default quota of 10,000 general purpose buckets. You can increase the bucket quota for an account by submitting a quota increase request. To request an increase for general purpose buckets, visit the Service Quotas console.

To manage your bucket-per-use pattern and simplify your infrastructure management, you can use AWS CloudFormation. You can create a custom CloudFormation template for your pattern that already defines all of your desired settings for your S3 general purpose buckets so that you can easily deploy and track any changes to your infrastructure. For more information, see AWS::S3::Bucket in the *AWS CloudFormation User Guide*.



When building a workload with a bucket-per-use pattern, we recommend that you create the buckets in your account regional namespace. By creating buckets in your account regional namespace, you avoid competing for bucket names against others and have assurance that only your account can ever create buckets with your selected naming convention. For more information on account regional namespaces, see Namespaces for general purpose buckets.

General purpose bucket naming rules

When you create a general purpose bucket, make sure that you consider the length, valid characters, formatting, and uniqueness of bucket names. The following sections provide information about general purpose bucket naming, including naming rules, best practices, and an example for how you can create buckets in your account regional namespace. an example for creating a general purpose bucket with a name that includes a globally unique identifier (GUID).

For information about object key names, see Creating object key names.

To create a general purpose bucket, see the section called “Creating a general purpose bucket”.

Topics

- General purpose buckets naming rules
- Account regional namespace naming rules
- Example general purpose bucket names
- Best practices
- Creating a bucket that uses a GUID in the bucket name
- Creating a bucket in your account regional namespace

General purpose buckets naming rules

The following naming rules apply for general purpose buckets.

- Bucket names must be between 3 (min) and 63 (max) characters long.
- Bucket names can consist only of lowercase letters, numbers, periods (`.`), and hyphens (`-`).
- Bucket names must begin and end with a letter or number.
- Bucket names must not contain two adjacent periods.
- Bucket names must not be formatted as an IP address (for example, `192.168.5.4`).
- Bucket names must not start with the prefix `xn--`.
- Bucket names must not start with the prefix `sthree-`.
- Bucket names must not start with the prefix `amzn-s3-demo-`.
- Bucket names must not end with the suffix `-s3alias`. This suffix is reserved for access point alias names. For more information, see Access point aliases.
- Bucket names must not end with the suffix `--ol-s3`. This suffix is reserved for Object Lambda Access Point alias names. For more information, see How to use a bucket-style alias for your S3 bucket Object Lambda Access Point.
- Bucket names must not end with the suffix `.mrap`. This suffix is reserved for Multi-Region Access Point names. For more information, see Rules for naming Amazon S3 Multi-Region Access Points.
- Bucket names must not end with the suffix `--x-s3`. This suffix is reserved for directory buckets. For more information, see Directory bucket naming rules.
- Bucket names must not end with the suffix `--table-s3`. This suffix is reserved for S3 Tables buckets. For more information, see Amazon S3 table bucket, table, and namespace naming rules.
- Bucket names can only end with the suffix `-an` when you are creating buckets in your account regional namespace. For more information, see Namespaces for general purpose buckets.

- Buckets used with Amazon S3 Transfer Acceleration can't have periods (`.`) in their names. For more information about Transfer Acceleration, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

Important

- General purpose buckets exist in a global namespace, which means that each bucket name must be unique across all AWS accounts in all the AWS Regions within a partition. A partition is a grouping of Regions. AWS currently has four partitions: `aws` (Standard Regions), `aws-cn` (China Regions), `aws-us-gov` (AWS GovCloud (US)), and `aws-eusc` (European Sovereign Cloud). After creating a general purpose bucket in the shared global namespace, that bucket name is unavailable for anyone else to create within a partition. When a bucket owner deletes their bucket, the bucket name might become available again in the global namespace for anyone to re-create. However, the name might not become available immediately, and in some cases might not become available again at all.
- A bucket name in the shared global namespace can't be used by another AWS account in the same partition until the bucket is deleted. **After you delete a bucket in the shared global namespace, be aware that another AWS account in the same partition can use the same bucket name for a new bucket and can therefore potentially receive requests intended for the deleted bucket.** If you want to prevent this, or if you want to continue to use the same bucket name, don't delete the bucket. We recommend that you empty the bucket and keep it, and instead, block any bucket requests as needed. For buckets no longer in active use, we recommend emptying the bucket of all objects to minimize costs while retaining the bucket itself.
- We recommend creating buckets in your account regional namespace for assurance that only your account can ever own these bucket names.
- When you create a general purpose bucket, you choose its name and the AWS Region to create it in. After you create a general purpose bucket, you can't change its name or Region.
- Don't include sensitive information in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

Note

Before March 1, 2018, buckets created in the US East (N. Virginia) Region could have names that were up to 255 characters long and included uppercase letters and underscores. Beginning March 1, 2018, new buckets in US East (N. Virginia) must conform to the same rules applied in all other Regions.

Account regional namespace naming rules

Although Amazon S3 general purpose buckets exist in a shared global namespace, you can optionally create buckets in your account regional namespace. The account regional namespace is a reserved subdivision of the global bucket namespace where only your account can create general purpose buckets. New general purpose buckets created in your account regional namespace are unique to your account and can never be re-created by another account. These buckets support all the S3 features and AWS services that general purpose buckets in the shared global namespace already support, your applications require no change to interact with buckets in your account regional namespace.

General purpose buckets in your account regional namespace must follow a specific naming convention. These buckets consist of a bucket name prefix that you create, and a suffix that contains your 12-digit AWS account ID, the AWS Region code, and ends with `-an`.

```
bucket-name-prefix-accountId-region-an
```

For example, the following general purpose bucket exists in the account regional namespace for AWS account 111122223333 in the us-west-2 Region:

```
amzn-s3-demo-bucket-111122223333-us-west-2-an
```

To create bucket in your account regional namespace, you make a `CreateBucket` request and specify the `x-amz-bucket-namespace` request header with the value set to `account-regional` along with specifying an account regional namespace formatted bucket name: *`customer-chosen-name`*`-AWS-Account-ID-AWS-Region-an`. For example, you could specify to create a bucket named: *`amzn-s3-demo-bucket`*`-111122223333-us-east-1-an` where your account regional suffix is `-111122223333-us-east-1-an`. For more information on account regional namespaces, see Namespaces for general purpose buckets.

Example general purpose bucket names

The following bucket names show examples of which characters are allowed in general purpose bucket names: a-z, 0-9, and hyphens (-). The `amzn-s3-demo-` reserved prefix is used here only for illustration. Because it's a reserved prefix, you can't create bucket names that start with `amzn-s3-demo-`.

- `amzn-s3-demo-bucket1-a1b2c3d4-5678-90ab-cdef-example11111`
- `amzn-s3-demo-bucket`

The following examples shows bucket names in your account regional namespace. These buckets must adhere to the specific account regional namespace naming convention: *`customer-chosen-name`*`-AWS-Account-ID-AWS-Region-an`

- `amzn-s3-demo-bucket-111122223333-us-west-2-an`
- `amzn-s3-demo-bucket-012345678910-ap-southeast-2-an`

The following example bucket names are valid but not recommended for uses other than static website hosting because they contain periods (`.`):

- `example.com`
- `www.example.com`
- `my.example.s3.bucket`

The following example bucket names are *not* valid:

- `amzn_s3_demo_bucket` (contains underscores)
- `AmznS3DemoBucket` (contains uppercase letters)
- `amzn-s3-demo-bucket-` (starts with `amzn-s3-demo-` prefix and ends with a hyphen)
- `example..com` (contains two periods in a row)
- `192.168.5.4` (matches format of an IP address)

Best practices

When naming your general purpose buckets, consider the following bucket naming best practices.

Create buckets in your account regional namespace

We recommend creating buckets in your account regional namespace for assurance that only your account can ever own these bucket names. With account regional namespaces, you can create predictable bucket names across multiple AWS Regions with assurance that no other account can create bucket names in your namespace.

Choose a bucket naming scheme that's unlikely to cause naming conflicts

If your application automatically creates buckets, choose a bucket naming scheme that's unlikely to cause naming conflicts. Ensure that your application logic will choose a different bucket name if a bucket name is already taken.

Append globally unique identifiers (GUIDs) to bucket names

We recommend that you create bucket names that aren't predictable. Don't write code assuming your chosen bucket name is available unless you have already created the bucket. One method for creating bucket names that aren't predictable is to append a Globally Unique Identifier (GUID) to your bucket name, for example, `amzn-s3-demo-bucket-a1b2c3d4-5678-90ab-cdef-example11111`. For more information, see the section called “Creating a bucket that uses a GUID in the bucket name”.

Avoid using periods (`.`) in bucket names

For best compatibility, we recommend that you avoid using periods (`.`) in bucket names, except for buckets that are used only for static website hosting. If you include periods in a bucket's name, you can't use virtual-host-style addressing over HTTPS, unless you perform your own certificate validation. The security certificates used for virtual hosting of buckets don't work for buckets with periods in their names.

This limitation doesn't affect buckets used for static website hosting, because static website hosting is available only over HTTP. For more information about virtual-host-style addressing, see Virtual hosting of general purpose buckets. For more information about static website hosting, see Hosting a static website using Amazon S3.

Choose a relevant name

When you name a bucket, we recommend that you choose a name that's relevant to you or your business. Avoid using names associated with others. For example, avoid using `AWS` or `Amazon` in your bucket name.

Don't delete buckets so that you can reuse bucket names

If a bucket is empty, you can delete it. After a bucket is deleted, the name becomes available for reuse. However, you aren't guaranteed to be able to reuse the name right away, or at all. After you delete a bucket in the shared global namespace, some time might pass before you can reuse the name. In addition, another AWS account might create a bucket with the same name before you can reuse the name.

After you delete a general purpose bucket in the shared global namespace, be aware that another AWS account in the same partition can use the same general purpose bucket name for a new bucket and can therefore potentially receive requests intended for the deleted general purpose bucket. If you want to prevent this, or if you want to continue to use the same general purpose bucket name, don't delete the general purpose bucket. We recommend that you empty the bucket and keep it, and instead, block any bucket requests as needed.

Creating a bucket that uses a GUID in the bucket name

The following examples show you how to create a general purpose bucket that uses a GUID at the end of the bucket name.

Using the AWS CLI

The following AWS CLI example creates a general purpose bucket in the US West (N. California) Region (`us-west-1`) Region with an example bucket name that uses a globally unique identifier (GUID). To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket1$(uuidgen | tr -d - | tr '[:upper:]' '[:lower:]' ) \
    --region us-west-1 \
    --create-bucket-configuration LocationConstraint=us-west-1
```

Using the AWS SDK for Java

The following example shows you how to create a with a GUID at the end of the bucket name in US East (N. Virginia) Region (`us-east-1`) by using the AWS SDK for Java. To use this example, replace the *`user input placeholders`* with your own information. For information about other AWS SDKs, see Tools to Build on AWS.

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
```

```
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.Bucket;
import com.amazonaws.services.s3.model.CreateBucketRequest;

import java.util.List;
import java.util.UUID;

public class CreateBucketWithUUID {
    public static void main(String[] args) {
        final AmazonS3 s3 =
 AmazonS3ClientBuilder.standard().withRegion(Regions.US_EAST_1).build();
        String bucketName = "amzn-s3-demo-bucket" +
 UUID.randomUUID().toString().replace("-", "");
        CreateBucketRequest createRequest = new CreateBucketRequest(bucketName);
        System.out.println(bucketName);
        s3.createBucket(createRequest);
    }
}
```

Creating a bucket in your account regional namespace

The following examples show you how to create a general purpose bucket in your account regional namespace.

Using the AWS CLI

The following AWS CLI example creates a general purpose bucket in the account regional namespace for AWS account 012345678910 in the US West (N. California) Region (`us-west-1`) Region. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket-012345678910-us-west-1-an \
    --bucket-namespace account-regional
    --region us-west-1 \
    --create-bucket-configuration LocationConstraint=us-west-1
```

General purpose bucket quotas, limitations, and restrictions

An Amazon S3 general purpose bucket is owned by the AWS account that created it. Bucket ownership is not transferable to another account.

Bucket quotas

By default, you can create up to 10,000 general purpose buckets per AWS account. To request a quota increase for general purpose buckets, visit the Service Quotas console.

Important

We strongly recommend using only paginated `ListBuckets` requests. Unpaginated `ListBuckets` requests are only supported for AWS accounts set to the default general purpose bucket quota of 10,000. If you have an approved general purpose bucket quota above 10,000, you must send paginated `ListBuckets` requests to list your account's buckets. All unpaginated `ListBuckets` requests will be rejected for AWS accounts with a general purpose bucket quota greater than 10,000.

Note

You must use the following AWS Regions to view your quota, bucket utilization, or request an increase for your general purpose buckets in your AWS account.

- General purpose bucket quotas for commercial Regions can only be viewed and managed from US East (N. Virginia).
- General purpose bucket quotas for AWS GovCloud (US) can only be viewed and managed from AWS GovCloud (US-West).

For information about service quotas, see AWS service quotas in the *Amazon Web Services General Reference*.

Objects and bucket limitations

There is no max bucket size or limit to the number of objects that you can store in a bucket. You can store all of your objects in a single bucket, or you can organize them across several buckets. However, you can't create a bucket from within another bucket.

Bucket naming rules

When you create a bucket, you choose its name and the AWS Region to create it in. After you create a bucket, you can't change its name or Region. For more information about bucket naming, see General purpose bucket naming rules.

Accessing an Amazon S3 general purpose bucket

You can access your Amazon S3 general purpose buckets by using the Amazon S3 console, AWS Command Line Interface, AWS SDKs, or the Amazon S3 REST API. Each method of accessing an S3 general purpose bucket supports specific use cases. For more information, see the following sections.

Topics

- Use cases
- Amazon S3 console
- AWS CLI
- AWS SDKs
- Amazon S3 REST API
- Virtual hosting of general purpose buckets

Use cases

Depending on the use case for your Amazon S3 general purpose bucket, there are different recommended methods to access the underlying data in your buckets. The following list includes common use cases for accessing your data.

- **Static websites** – You can use Amazon S3 to host a static website. In this use case, you can configure your S3 general purpose bucket to function like a website. For an example that walks you through the steps of hosting a website on Amazon S3, see Tutorial: Configuring a static website on Amazon S3.

  To host a static website with security settings like Block Public Access enabled, we recommend using Amazon CloudFront with Origin Access Control (OAC) and implementing additional security headers, such as HTTPS. For more information, see Getting started with a secure static website.

> **Note**
>
> Amazon S3 supports both virtual-hosted–style and path-style URLs for static website access. Because buckets can be accessed using path-style and virtual-hosted–style URLs, we recommend that you create buckets with DNS-compliant bucket names. For more information, see General purpose bucket quotas, limitations, and restrictions.

- **Shared datasets** – As you scale on Amazon S3, it's common to adopt a multi-tenant model, where you assign different end customers or business units to unique prefixes within a shared general purpose bucket. By using Amazon S3 access points, you can divide one large bucket policy into separate, discrete access point policies for each application that needs to access the shared dataset. This approach makes it simpler to focus on building the right access policy for an application without disrupting what any other application is doing within the shared dataset. For more information, see Managing access to shared datasets with access points.
- **Browsing and managing data** – Storage Browser for Amazon S3 is an open source component that you can add to your web applications to provide your end users with a simple interface for browsing and managing Amazon S3 data. For more information, see Working with Storage Browser for Amazon S3.
- **High-throughput workloads** – Mountpoint for Amazon S3 is a high-throughput open source file client for mounting an Amazon S3 general purpose bucket as a local file system. With Mountpoint, your applications can access objects stored in Amazon S3 through file-system operations, such as open and read. Mountpoint automatically translates these operations into S3 object API calls, giving your applications access to the elastic storage and throughput of Amazon S3 through a file interface. For more information, see Mount an Amazon S3 bucket as a local file system.
- **Multi-Region applications** – Amazon S3 Multi-Region Access Points provide a global endpoint that applications can use to fulfill requests from S3 general purpose buckets that are located in multiple AWS Regions. You can use Multi-Region Access Points to build multi-Region applications with the same architecture that's used in a single Region, and then run those applications anywhere in the world. Instead of sending requests over the public internet, Multi-Region Access Points provide built-in network resilience with acceleration of internet-based requests to Amazon S3. For more information, see Managing multi-Region traffic with Multi-Region Access Points.
- **Secure Shell (SSH) File Transfer Protocol (SFTP)** – If you're trying to securely transfer sensitive data over the internet, you can use an SFTP-enabled server with your Amazon S3 general purpose bucket. AWS SFTP is a network protocol that supports the full security and

authentication functionality of SSH. With this protocol, you have fine-grained control over user identity, permissions, and keys or you can use IAM policies to manage access. To associate an SFTP enabled server with your Amazon S3 bucket, make sure to create your SFTP-enabled server first. Then, you set up user accounts, and associate the server with an Amazon S3 general purpose bucket. For a walkthrough of this process, see AWS Transfer for SFTP – Fully Managed SFTP Service for Amazon S3 in *AWS Blogs*.

Amazon S3 console

The console is a web-based user interface for managing Amazon S3 and AWS resources. With the Amazon S3 console, you can easily access a bucket and modify the bucket's properties. You can also perform most bucket operations by using the console UI, without having to write any code.

If you've signed up for an AWS account, you can access the Amazon S3 console by signing into the Amazon S3 console and choosing **S3** from the Amazon S3 console home page. You can also use this link to directly access the https://console.aws.amazon.com/s3/.

AWS CLI

You can use the AWS CLI to issue commands or build scripts at your system's command line to perform AWS (including S3) tasks. For example, if you need to access multiple buckets, you can save time by using the AWS CLI to automate common and repetitive tasks. Scriptability and repeatability for common actions are frequent considerations as organizations scale.

The AWS CLI provides commands for a broad set of AWS services. The AWS CLI is supported on Windows, macOS, and Linux. To get started, see the *AWS Command Line Interface User Guide*. For more information about the commands for Amazon S3, see s3api and s3control in the *AWS CLI Command Reference*.

AWS SDKs

AWS provides SDKs (software development kits) that consist of libraries and sample code for various programming languages and platforms (Java, Python, Ruby, .NET, iOS, Android, and so on). The AWS SDKs provide a convenient way to create programmatic access to S3 and AWS. Amazon S3 is a REST service. You can send requests to Amazon S3 using the AWS SDK libraries, which wrap the underlying Amazon S3 REST API and simplify your programming tasks. For example, the SDKs take care of tasks such as calculating signatures, cryptographically signing requests, managing

errors, and retrying requests automatically. For information about the AWS SDKs, including how to download and install them, see Tools for AWS.

Every interaction with Amazon S3 is either authenticated or anonymous. If you are using the AWS SDKs, the libraries compute the signature for authentication from the keys that you provide. For more information about how to make requests to Amazon S3, see Making requests .

Amazon S3 REST API

The architecture of Amazon S3 is designed to be programming language-neutral, using AWS-supported interfaces to store and retrieve objects. You can access S3 and AWS programmatically by using the Amazon S3 REST API. The REST API is an HTTP interface to Amazon S3. With the REST API, you use standard HTTP requests to create, fetch, and delete buckets and objects.

To use the REST API, you can use any toolkit that supports HTTP. You can even use a browser to fetch objects, as long as they are anonymously readable.

The REST API uses standard HTTP headers and status codes, so that standard browsers and toolkits work as expected. In some areas, we have added functionality to HTTP (for example, we added headers to support access control). In these cases, we have done our best to add the new functionality in a way that matches the style of standard HTTP usage.

If you make direct REST API calls in your application, you must write the code to compute the signature and add it to the request. For more information about how to make requests to Amazon S3, see Making requests in the *Amazon S3 API Reference*.

Virtual hosting of general purpose buckets

Virtual hosting is the practice of serving multiple websites from a single web server. One way to differentiate sites in your Amazon S3 REST API requests is by using the apparent hostname of the Request-URI instead of just the path name part of the URI. An ordinary Amazon S3 REST request specifies a bucket by using the first slash-delimited component of the Request-URI path. Instead, you can use Amazon S3 virtual hosting to address a general purpose bucket in a REST API call by using the HTTP `Host` header. In practice, Amazon S3 interprets `Host` as meaning that most buckets are automatically accessible for limited types of requests at `https://`*`bucket-name`*`.s3.`*`region-code`*`.amazonaws.com`. For a complete list of Amazon S3 Regions and endpoints, see Amazon S3 endpoints and quotas in the *Amazon Web Services General Reference*.

Virtual hosting also has other benefits. By naming your bucket after your registered domain name and by making that name a DNS alias for Amazon S3, you can completely customize the

URL of your Amazon S3 resources, for example, `http://my.bucket-name.com/`. You can also publish to the "root directory" of your bucket's virtual server. This ability can be important because many existing applications search for files in this standard location. For example, `favicon.ico`, `robots.txt`, and `crossdomain.xml` are all expected to be found at the root.

> **Important**
>
> When you're using virtual-hosted–style general purpose buckets with SSL, the SSL wildcard certificate matches only buckets that do not contain dots (`.`). To work around this limitation, use HTTP or write your own certificate-verification logic. For more information, see Amazon S3 Path Deprecation Plan on the *AWS News Blog*.

Topics

- Path-style requests
- Virtual-hosted–style requests
- HTTP Host header bucket specification
- Examples
- Customizing Amazon S3 URLs with CNAME records
- How to associate a hostname with an Amazon S3 bucket
- Limitations
- Backward compatibility

Path-style requests

Currently, Amazon S3 supports both virtual-hosted-style and path-style URL access in all AWS Regions.

In Amazon S3, path-style URLs use the following format:

```
https://s3.region-code.amazonaws.com/bucket-name/key-name
```

For example, if you create a bucket named `amzn-s3-demo-bucket1` in the US West (Oregon) Region, and you want to access the puppy `.jpg` object in that bucket, you can use the following path-style URL:

```
https://s3.us-west-2.amazonaws.com/amzn-s3-demo-bucket1/puppy.jpg
```

Important

Update (September 23, 2020) – To make sure that customers have the time that they need to transition to virtual-hosted–style URLs, we have decided to delay the deprecation of path-style URLs. For more information, see Amazon S3 Path Deprecation Plan – The Rest of the Story in the *AWS News Blog*.

Warning

When hosting website content that will be accessed from a web browser, avoid using path-style URLs, which might interfere with the browser same origin security model. To host website content, we recommend that you use either S3 website endpoints or a CloudFront distribution. For more information, see Website endpoints and Deploy a React-based single-page application to Amazon S3 and CloudFront in the *AWS Perspective Guidance Patterns*.

Virtual-hosted–style requests

In a virtual-hosted–style URI, the bucket name is part of the domain name in the URL.

Amazon S3 virtual-hosted–style URLs use the following format:

```
https://bucket-name.s3.region-code.amazonaws.com/key-name
```

In this example, `amzn-s3-demo-bucket1` is the bucket name, US West (Oregon) is the Region, and `puppy.png` is the key name:

```
https://amzn-s3-demo-bucket1.s3.us-west-2.amazonaws.com/puppy.png
```

HTTP `Host` header bucket specification

As long as your `GET` request does not use the SSL endpoint, you can specify the bucket for the request by using the HTTP `Host` header. The `Host` header in a REST request is interpreted as follows:

- If the `Host` header is omitted or its value is `s3.region-code.amazonaws.com`, the bucket for the request will be the first slash-delimited component of the Request-URI, and the key for the request will be the rest of the Request-URI. This is the ordinary method, as illustrated by the first and second examples in this section. Omitting the `Host` header is valid only for HTTP 1.0 requests.
- Otherwise, if the value of the `Host` header ends in `.s3.region-code.amazonaws.com`, the bucket name is the leading component of the `Host` header's value up to `.s3.region-code.amazonaws.com`. The key for the request is the Request-URI. This interpretation exposes buckets as subdomains of `.s3.region-code.amazonaws.com`, as illustrated by the third and fourth examples in this section.
- Otherwise, the bucket for the request is the lowercase value of the `Host` header, and the key for the request is the Request-URI. This interpretation is useful when you have registered the same DNS name as your bucket name and have configured that name to be a canonical name (CNAME) alias for Amazon S3. The procedure for registering domain names and configuring CNAME DNS records is beyond the scope of this guide, but the result is illustrated by the final example in this section.

Examples

This section provides example URLs and requests.

Example– Path-style URLs and requests

This example uses the following:

- Bucket Name - `example.com`
- Region - US East (N. Virginia)
- Key Name - `homepage.html`

The URL is as follows:

```
http://s3.us-east-1.amazonaws.com/example.com/homepage.html
```

The request is as follows:

```
GET /example.com/homepage.html HTTP/1.1
```

```
Host: s3.us-east-1.amazonaws.com
```

The request with HTTP 1.0 and omitting the `Host` header is as follows:

```
GET /example.com/homepage.html HTTP/1.0
```

For information about DNS-compatible names, see Limitations. For more information about keys, see Keys.

Example– Virtual-hosted–style URLs and requests

This example uses the following:

- **Bucket name** - `amzn-s3-demo-bucket1`
- **Region** - Europe (Ireland)
- **Key name** - `homepage.html`

The URL is as follows:

```
http://amzn-s3-demo-bucket1.s3.eu-west-1.amazonaws.com/homepage.html
```

The request is as follows:

```
GET /homepage.html HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.eu-west-1.amazonaws.com
```

Example– CNAME alias method

To use this method, you must configure your DNS name as a CNAME alias for *`bucket-name`*`.s3.us-east-1.amazonaws.com`. For more information, see Customizing Amazon S3 URLs with CNAME records.

This example uses the following:

- Bucket Name - `example.com`
- **Key name** - `homepage.html`

The URL is as follows:

```
http://www.example.com/homepage.html
```

The example is as follows:

```
GET /homepage.html HTTP/1.1
Host: www.example.com
```

Customizing Amazon S3 URLs with CNAME records

Depending on your needs, you might not want `s3.region-code.amazonaws.com` to appear on your website or service. For example, if you're hosting website images on Amazon S3, you might prefer `http://images.example.com/` instead of `http://images.example.com.s3.us-east-1.amazonaws.com/`. Any bucket with a DNS-compatible name can be referenced as follows: `http://BucketName.s3.Region.amazonaws.com/[Filename]`, for example, `http://images.example.com.s3.us-east-1.amazonaws.com/mydog.jpg`. By using CNAME, you can map `images.example.com` to an Amazon S3 hostname so that the previous URL could become `http://images.example.com/mydog.jpg`.

Your bucket name must be the same as the CNAME. For example, if you create a CNAME to map `images.example.com` to `images.example.com.s3.us-east-1.amazonaws.com`, both `http://images.example.com/filename` and `http://images.example.com.s3.us-east-1.amazonaws.com/filename` will be the same.

The CNAME DNS record should alias your domain name to the appropriate virtual hosted–style hostname. For example, if your bucket name and domain name are `images.example.com` and your bucket is in the US East (N. Virginia) Region, the CNAME record should alias to `images.example.com.s3.us-east-1.amazonaws.com`.

```
images.example.com CNAME    images.example.com.s3.us-east-1.amazonaws.com.
```

Amazon S3 uses the hostname to determine the bucket name. So the CNAME and the bucket name must be the same. For example, suppose that you have configured `www.example.com` as a CNAME for `www.example.com.s3.us-east-1.amazonaws.com`. When you access `http://www.example.com`, Amazon S3 receives a request similar to the following:

Example

```
GET / HTTP/1.1
Host: www.example.com
```

```
Date: date
Authorization: signatureValue
```

Amazon S3 sees only the original hostname `www.example.com` and is unaware of the CNAME mapping used to resolve the request.

You can use any Amazon S3 endpoint in a CNAME alias. For example, `s3.ap-southeast-1.amazonaws.com` can be used in CNAME aliases. For more information about endpoints, see Request endpoints in the *Amazon S3 API Reference*. To create a static website by using a custom domain, see Tutorial: Configuring a static website using a custom domain registered with Route 53.

> **Important**
>
> When using custom URLs with CNAMEs, you will need to ensure a matching bucket exists for any CNAME or alias record you configure. For example, if you create DNS entries for `www.example.com` and `login.example.com` to publish web content using S3, you will need to create both buckets `www.example.com` and `login.example.com`.
> When a CNAME or alias records is configured pointing to an S3 endpoint without a matching bucket, any AWS user can create that bucket and publish content under the configured alias, even if ownership is not the same.
> For the same reason, we recommend that you change or remove the corresponding CNAME or alias when deleting a bucket.

How to associate a hostname with an Amazon S3 bucket

To associate a hostname with an Amazon S3 bucket by using a CNAME alias

1. Select a hostname that belongs to a domain that you control.

   This example uses the `images` subdomain of the `example.com` domain.
2. Create a bucket that matches the hostname.

   In this example, the host and bucket names are `images.example.com`. The bucket name must *exactly* match the hostname.
3. Create a CNAME DNS record that defines the hostname as an alias for the Amazon S3 bucket.

   For example:

```
images.example.com CNAME images.example.com.s3.us-west-2.amazonaws.com
```

> **Important**
>
> For request-routing reasons, the CNAME DNS record must be defined exactly as shown in the preceding example. Otherwise, it might appear to operate correctly, but it will eventually result in unpredictable behavior.

The procedure for configuring CNAME DNS records depends on your DNS server or DNS provider. For specific information, see your server documentation or contact your provider.

Limitations

SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the AWS SDKs.

Backward compatibility

The following sections cover various aspects of Amazon S3 backward compatibility that relate to path-style and virtual-hosted–style URL requests.

Legacy endpoints

Some Regions support legacy endpoints. You might see these endpoints in your server access logs or AWS CloudTrail logs. For more information, review the following information. For a complete list of Amazon S3 Regions and endpoints, see Amazon S3 endpoints and quotas in the *Amazon Web Services General Reference*.

> **Important**
>
> Although you might see legacy endpoints in your logs, we recommend that you always use the standard endpoint syntax to access your buckets.
> Amazon S3 virtual-hosted–style URLs use the following format:
>
> ```
> https://bucket-name.s3.region-code.amazonaws.com/key-name
> ```
>
> In Amazon S3, path-style URLs use the following format:

```
https://s3.region-code.amazonaws.com/bucket-name/key-name
```

s3-Region

Some older Amazon S3 Regions support endpoints that contain a dash (-) between s3 and the Region code (for example, s3-us-west-2), instead of a dot (for example, s3.us-west-2). If your bucket is in one of these Regions, you might see the following endpoint format in your server access logs or CloudTrail logs:

```
https://bucket-name.s3-region-code.amazonaws.com
```

In this example, the bucket name is amzn-s3-demo-bucket1 and the Region is US West (Oregon):

```
https://amzn-s3-demo-bucket1.s3-us-west-2.amazonaws.com
```

Legacy global endpoint

For some Regions, you can use the legacy global endpoint to construct requests that do not specify a Region-specific endpoint. The legacy global endpoint point is as follows:

```
bucket-name.s3.amazonaws.com
```

In your server access logs or CloudTrail logs, you might see requests that use the legacy global endpoint. In this example, the bucket name is amzn-s3-demo-bucket1 and the legacy global endpoint is:

```
https://amzn-s3-demo-bucket1.s3.amazonaws.com
```

Virtual-hosted–style requests for US East (N. Virginia)

Requests made with the legacy global endpoint go to the US East (N. Virginia) Region by default. Therefore, the legacy global endpoint is sometimes used in place of the Regional endpoint for US East (N. Virginia). If you create a bucket in US East (N. Virginia) and use the global endpoint, Amazon S3 routes your request to this Region by default.

Virtual-hosted–style requests for other Regions

The legacy global endpoint is also used for virtual-hosted–style requests in other supported Regions. If you create a bucket in a Region that was launched before March 20, 2019, and use the legacy global endpoint, Amazon S3 updates the DNS record to reroute the request to the correct location, which might take time. In the meantime, the default rule applies, and your virtual-hosted–style request goes to the US East (N. Virginia) Region. Amazon S3 then redirects it with an HTTP 307 Temporary Redirect to the correct Region.

For S3 buckets in Regions launched after March 20, 2019, the DNS server doesn't route your request directly to the AWS Region where your bucket resides. It returns an HTTP 400 Bad Request error instead. For more information, see Making requests in the *Amazon S3 API Reference*.

Path-style requests

For the US East (N. Virginia) Region, you can use the legacy global endpoint for path-style requests.

For all other Regions, the path-style syntax requires that you use the Region-specific endpoint when attempting to access a bucket. If you try to access a bucket with the legacy global endpoint or another endpoint that is different than the one for the Region where the bucket resides, you receive an HTTP response code 301 Permanent Redirect error and a message that indicates the correct URI for your resource. For example, if you use `https://s3.amazonaws.com/bucket-name` for a bucket that was created in the US West (Oregon) Region, you will receive an HTTP 301 Permanent Redirect error.

Creating a general purpose bucket

To upload your data to Amazon S3, you must first create an Amazon S3 general purpose bucket in one of the AWS Regions. The AWS account that creates the bucket owns it. When you create a bucket, you must choose a bucket name and Region. During the creation process, you can optionally choose other storage management options for the bucket.

> **Important**
>
> After you create a bucket, you can't change the bucket name, the bucket owner, or the Region. For more information about bucket naming, see the section called “Naming rules”.

By default, you can create up to 10,000 general purpose buckets per AWS account. To request a quota increase for general purpose buckets, visit the Service Quotas console.

You can store any number of objects in a bucket. For a list of restriction and limitations related to Amazon S3 general purpose buckets, see General purpose bucket quotas, limitations, and restrictions.

General purpose bucket settings

When you are creating a general purpose bucket, you must decide if you want to create a global general purpose bucket or a general purpose bucket in your account regional namespace. This decision along with the bucket name and region cannot be changed after creation.

When you're creating a general purpose bucket, you can use the following settings to control various aspects of your bucket's behavior:

- **S3 Bucket Namespace** – By default, Amazon S3 general purpose buckets exist in a global namespace. When you create a general purpose bucket, you can choose to create a bucket in the shared global namespace or you can choose to create a bucket in your account regional namespace. Your account regional namespace is a subdivision of the global namespace that only your account can create buckets in. New general purpose buckets created in your account regional namespace are unique to your account and can never be re-created by another account. These buckets support all the S3 features and AWS services that general purpose buckets in the shared global namespace already support, your applications require no change to interact with buckets in your account regional namespace. For more information on bucket namespaces, see Namespaces for general purpose buckets.
- **S3 Object Ownership** – S3 Object Ownership is an Amazon S3 bucket-level setting that you can use both to control ownership of objects that are uploaded to your bucket and to disable or enable access control lists (ACLs). By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. With ACLs disabled, the bucket owner owns every object in the bucket and manages access to data exclusively by using policies. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.
- **S3 Object Lock** – S3 Object Lock can help prevent Amazon S3 objects from being deleted or overwritten for a fixed amount of time or indefinitely. Object Lock uses a *write-once-read-many* (WORM) model to store objects. You can use Object Lock to help meet regulatory requirements that require WORM storage, or to add another layer of protection against object changes or deletion. For more information, see the section called “Locking objects”.
- **Tags** – An AWS tag is a key-value pair that holds metadata. You attach the tags to your Amazon S3 resources, such as buckets. You can tag resources when you create them or manage tags on existing resources. You can use tags for cost allocation to track storage costs by bucket tag in

AWS Billing and Cost Management. You can also use tags for attribute-based access control (ABAC), to scale access permissions and grant access to S3 resources based on their tags. For more information, see Using tags with S3 general purpose buckets.

After you create a general purpose bucket, or when you're creating a general purpose bucket by using the Amazon S3 console, you can also use the following settings to control other aspects of your bucket's behavior:

- **S3 Block Public Access** – The S3 Block Public Access feature provides settings for access points, buckets, and accounts to help you manage public access to Amazon S3 resources. By default, new buckets, access points, and objects don't allow public access. However, users can modify bucket policies, access point policies, or object permissions to allow public access. S3 Block Public Access settings override these policies and permissions so that you can limit public access to these resources. For more information, see the section called “Blocking public access”.
- **S3 Versioning** – Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your bucket. With versioning, you can easily recover from both unintended user actions and application failures. By default, versioning is disabled for buckets. For more information, see the section called “Retaining multiple versions of objects”.
- **Default encryption** – You can set the default encryption type for all objects in your bucket. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the base level of encryption configuration for every bucket in Amazon S3. All new objects uploaded to an S3 bucket are automatically encrypted with SSE-S3 as the base level of encryption. If you want to use a different type of default encryption, you can specify server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C) to encrypt your data. For more information, see Setting default server-side encryption behavior for Amazon S3 buckets.

You can use the Amazon S3 console, Amazon S3 REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs to create a general purpose bucket. For more information about the permissions required to create a general purpose bucket, see CreateBucket in the *Amazon Simple Storage Service API Reference*.

If you're having trouble creating an Amazon S3 bucket, see How do I troubleshoot errors when creating an Amazon S3 bucket? on AWS re:Post.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

   Note

   - After you create a bucket, you can't change its Region.
   - To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.
3. In the left navigation pane, choose **General purpose buckets**.
4. Choose **Create bucket**. The **Create bucket** page opens.
5. For **Bucket name**, enter a name for your bucket.

   The bucket name must:

   - Be unique within a partition. A partition is a grouping of Regions. AWS currently has three partitions: `aws` (commercial Regions), `aws-cn` (China Regions), and `aws-us-gov` (AWS GovCloud (US) Regions).
   - Be between 3 and 63 characters long.
   - Consist only of lowercase letters, numbers, periods (`.`), and hyphens (`-`). For best compatibility, we recommend that you avoid using periods (`.`) in bucket names, except for buckets that are used only for static website hosting.
   - Begin and end with a letter or number.
   - For a complete list of bucket-naming rules, see General purpose bucket naming rules.

   Important

   - After you create the bucket, you can't change its name.

- Don't include sensitive information in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

6. (Optional) Under **General configuration**, you can choose to copy an existing bucket's settings to your new bucket. If you don't want to copy the settings of an existing bucket, skip to the next step.

> **Note**
>
> This option:
>
> - Isn't available in the AWS CLI and is only available in the Amazon S3 console
> - Doesn't copy the bucket policy from the existing bucket to the new bucket

   To copy an existing bucket's settings, under **Copy settings from existing bucket**, select **Choose bucket**. The **Choose bucket** window opens. Find the bucket with the settings that you want to copy, and select **Choose bucket**. The **Choose bucket** window closes, and the **Create bucket** window reopens.

   Under **Copy settings from existing bucket**, you now see the name of the bucket that you selected. The settings of your new bucket now match the settings of the bucket that you selected. If you want to remove the copied settings, choose **Restore defaults**. Review the remaining bucket settings on the **Create bucket** page. If you don't want to make any changes, you can skip to the final step.

7. Under **Object Ownership**, to disable or enable ACLs and control ownership of objects uploaded in your bucket, choose one of the following settings:

   ACLs disabled

   - **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the general purpose bucket. ACLs no longer affect access permissions to data in the S3 general purpose bucket. The bucket uses policies exclusively to define access control.

     By default, ACLs are disabled. A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except

in circumstances where you must control access for each object individually. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

ACLs enabled

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.

  If you apply the **Bucket owner preferred** setting, to require all Amazon S3 uploads to include the `bucket-owner-full-control` canned ACL, you can add a bucket policy that allows only object uploads that use this ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

> **Note**
>
> The default setting is **Bucket owner enforced**. To apply the default setting and keep ACLs disabled, only the `s3:CreateBucket` permission is needed. To enable ACLs, you must have the `s3:PutBucketOwnershipControls` permission.

8. Under **Block Public Access settings for this bucket**, choose the Block Public Access settings that you want to apply to the bucket.

   By default, all four Block Public Access settings are enabled. We recommend that you keep all settings enabled, unless you know that you need to turn off one or more of them for your specific use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

   > **Note**
   >
   > To enable all Block Public Access settings, only the `s3:CreateBucket` permission is required. To turn off any Block Public Access settings, you must have the `s3:PutBucketPublicAccessBlock` permission.

9. (Optional) By default, **Bucket Versioning** is disabled. Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your bucket. With versioning, you can recover

more easily from both unintended user actions and application failures. For more information about versioning, see Retaining multiple versions of objects with S3 Versioning.

To enable versioning on your bucket, choose **Enable**.

10. (Optional) Under **Tags**, you can choose to add tags to your bucket. With AWS cost allocation, you can use bucket tags to annotate billing for your use of a bucket. A tag is a key-value pair that represents a label that you assign to a bucket. For more information, see the section called "Using cost allocation tags".

    To add a bucket tag, enter a **Key** and optionally a **Value** and choose **Add Tag**.

11. To configure **Default encryption**, under **Encryption type**, choose one of the following:

    - **Server-side encryption with Amazon S3 managed keys (SSE-S3)**
    - **Server-side encryption with AWS Key Management Service keys (SSE-KMS)**
    - **Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)**

    > **Important**
    >
    > If you use the SSE-KMS or DSSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quota of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.

    Buckets and new objects are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption configuration. For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets. For more information about SSE-S3, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

    For more information about using server-side encryption to encrypt your data, see the section called "Data encryption".

12. If you chose **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** or **Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)**, do the following:

a. Under **AWS KMS key**, specify your KMS key in one of the following ways:

- To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

  Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
- To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that isn't listed, you must enter your KMS key ARN. If you want to use a KMS key that's owned by a different account, you must first have permission to use the key, and then you must enter the KMS key ARN. For more information about cross account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*. For more information about SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS). For more information about DSSE-KMS, see the section called "Dual-layer server-side encryption (DSSE-KMS)". When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

b. When you configure your bucket to use default encryption with SSE-KMS, you can also use S3 Bucket Keys. S3 Bucket Keys lower the cost of encryption by decreasing request traffic

from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys. S3 Bucket Keys aren't supported for DSSE-KMS.

By default, S3 Bucket Keys are enabled in the Amazon S3 console. We recommend leaving S3 Bucket Keys enabled to lower your costs. To disable S3 Bucket Keys for your bucket, under **Bucket Key**, choose **Disable**.

13. (Optional) S3 Object Lock helps protect new objects from being deleted or overwritten. For more information, see Locking objects with Object Lock. If you want to enable S3 Object Lock, do the following:

    a. Choose **Advanced settings**.

    > **Important**
    >
    > Enabling Object Lock automatically enables versioning for the bucket. After you've enabled and successfully created the bucket, you must also configure the Object Lock default retention and legal hold settings on the bucket's **Properties** tab.

    b. If you want to enable Object Lock, choose **Enable**, read the warning that appears, and acknowledge it.

    > **Note**
    >
    > To create an Object Lock enabled bucket, you must have the following permissions: `s3:CreateBucket`, `s3:PutBucketVersioning`, and `s3:PutBucketObjectLockConfiguration`.

14. Choose **Create bucket**.

Using the AWS SDKs

When you use the AWS SDKs to create a general purpose bucket, you must create a client and then use the client to send a request to create a bucket. As a best practice, you should create your client and bucket in the same AWS Region. If you don't specify a Region when you create a client or a bucket, Amazon S3 uses the default Region, US East (N. Virginia). If you want to constrain the bucket creation to a specific AWS Region, use the LocationConstraint condition key.

To create a client to access a dual-stack endpoint, you must specify an AWS Region. For more information, see Using Amazon S3 dual-stack endpoints in the *Amazon S3 API Reference*. For a list of available AWS Regions, see Amazon Simple Storage Service endpoints and quotas in the *AWS General Reference*.

When you create a client, the Region maps to the Region-specific endpoint. The client uses this endpoint to communicate with Amazon S3: `s3.region.amazonaws.com`. If your Region launched after March 20, 2019, your client and bucket must be in the same Region. However, you can use a client in the US East (N. Virginia) Region to create a bucket in any Region that launched before March 20, 2019. For more information, see Legacy endpoints.

These AWS SDK code examples perform the following tasks:

- **Create a client by explicitly specifying an AWS Region** – In the example, the client uses the `s3.us-west-2.amazonaws.com` endpoint to communicate with Amazon S3. You can specify any AWS Region. For a list of AWS Regions, see Regions and endpoints in the *AWS General Reference*.
- **Send a create bucket request by specifying only a bucket name** – The client sends a request to Amazon S3 to create the bucket in the Region where you created a client.
- **Retrieve information about the location of the bucket** – Amazon S3 stores bucket location information in the *location* subresource that's associated with the bucket.

For additional AWS SDK examples and examples in other languages, see Use CreateBucket with an AWS SDK or CLI in the *Amazon Simple Storage Service API Reference*.

Java

For examples of how to create a bucket with the AWS SDK for Java, see Create a bucket in the *Amazon S3 API Reference*.

When using the AWS SDK for Java v2, you can create a general purpose bucket with a globally unique identifier (GUID) appended to the bucket name to ensure uniqueness. Use `UUID.randomUUID().toString().replace("-", "")` to generate a GUID and concatenate it with your base bucket name. This approach helps avoid bucket naming conflicts across all AWS accounts.

.NET

For information about how to create and test a working sample, see the AWS SDK for .NET Version 3 API Reference.

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.S3.Util;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CreateBucketTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CreateBucketAsync().Wait();
        }

        static async Task CreateBucketAsync()
        {
            try
            {
                if (!(await AmazonS3Util.DoesS3BucketExistAsync(s3Client,
 bucketName)))
                {
                    var putBucketRequest = new PutBucketRequest
                    {
                        BucketName = bucketName,
                        UseClientRegion = true
                    };

                    PutBucketResponse putBucketResponse = await
 s3Client.PutBucketAsync(putBucketRequest);
```

```
                }
                // Retrieve the bucket location.
                string bucketLocation = await FindBucketLocationAsync(s3Client);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
        static async Task<string> FindBucketLocationAsync(IAmazonS3 client)
        {
            string bucketLocation;
            var request = new GetBucketLocationRequest()
            {
                BucketName = bucketName
            };
            GetBucketLocationResponse response = await
 client.GetBucketLocationAsync(request);
            bucketLocation = response.Location.ToString();
            return bucketLocation;
        }
    }
}
```

Ruby

For information about how to create and test a working sample, see the AWS SDK for Ruby - Version 3.

Example

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket
```

```
  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                  create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

  puts "Created bucket #{wrapper.bucket.name}."
  puts "Your bucket's region is: #{wrapper.location}"
end
```

```
run_demo if $PROGRAM_NAME == __FILE__
```

Using the AWS CLI

The following AWS CLI example creates a general purpose bucket in the US West (N. California) Region (`us-west-1`) Region with an example bucket name that uses a globally unique identifier (GUID). To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket1$(uuidgen | tr -d - | tr '[:upper:]' '[:lower:]' ) \
    --region us-west-1 \
    --create-bucket-configuration LocationConstraint=us-west-1
```

For more information and additional examples, see create-bucket in the *AWS CLI Command Reference*.

Viewing the properties for an S3 general purpose bucket

You can view properties for any Amazon S3 bucket you own. These settings include the following:

- **Bucket Versioning** – Keep multiple versions of an object in one general purpose bucket by using versioning. By default, versioning is disabled for a new bucket. For information about enabling versioning, see Enabling versioning on buckets.
- **Tags** – An AWS tag is a key-value pair that holds metadata. You attach the tags to your Amazon S3 resources, such as buckets. You can tag resources when you create them or manage tags on existing resources. You can use tags for cost allocation to track storage costs by bucket tag in AWS Billing and Cost Management. You can also use tags for attribute-based access control (ABAC), to scale access permissions and grant access to S3 resources based on their tags. For more information, see Using tags with S3 general purpose buckets.
- **Default encryption** – Enabling default encryption provides you with automatic server-side encryption. Amazon S3 encrypts an object before saving it to a disk and decrypts the object when you download it. For more information, see Setting default server-side encryption behavior for Amazon S3 buckets.
- **Server access logging** – Get detailed records for the requests that are made to your general purpose bucket with server access logging. By default, Amazon S3 doesn't collect server access

logs. For information about enabling server access logging, see Enabling Amazon S3 server access logging.

- **AWS CloudTrail data events** – Use CloudTrail to log data events. By default, trails don't log data events. Additional charges apply for data events. For more information, see Logging Data Events for Trails in the *AWS CloudTrail User Guide*.
- **Event notifications** – Enable certain Amazon S3 general purpose bucket events to send notification messages to a destination whenever the events occur. For more information, see Enabling and configuring event notifications using the Amazon S3 console.
- **Transfer acceleration** – Enable fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. For information about enabling transfer acceleration, see Enabling and using S3 Transfer Acceleration.
- **Object Lock** – Use S3 Object Lock to prevent an object from being deleted or overwritten for a fixed amount of time or indefinitely. For more information, see Locking objects with Object Lock.
- **Requester Pays** – Enable Requester Pays if you want the requester (instead of the general purpose bucket owner) to pay for requests and data transfers. For more information, see Using Requester Pays general purpose buckets for storage transfers and usage.
- **Static website hosting** – You can host a static website on Amazon S3. For more information, see Hosting a static website using Amazon S3.

You can view bucket properties using the AWS Management Console, AWS CLI, or AWS SDKs

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to view the properties for.
4. Choose the **Properties** tab.
5. On the **Properties** page, you can configure the above properties for the bucket.

Using the AWS CLI

View bucket properties with the AWS CLI

The following commands show how you can use the AWS CLI to list different general purpose bucket properties.

The following returns the tag set associated with the bucket *amzn-s3-demo-bucket1*. For more information about bucket tags see, Using cost allocation S3 bucket tags.

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket1
```

For more information and examples, see get-bucket-tagging in the *AWS CLI Command Reference*.

The following returns the versioning state of the bucket *amzn-s3-demo-bucket1*. For information about the bucket versioning, see Retaining multiple versions of objects with S3 Versioning.

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket1
```

For more information and examples, see get-bucket-versioning in the *AWS CLI Command Reference*.

The following returns the default encryption configuration for the bucket *amzn-s3-demo-bucket1*. By default, all buckets have a default encryption configuration that uses server-side encryption with Amazon S3 managed keys (SSE-S3). For information about the bucket default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets.

```
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket1
```

For more information and examples, see get-bucket-encryption in the *AWS CLI Command Reference*.

The following returns the notification configuration of the bucket *amzn-s3-demo-bucket1*. For information about the bucket event notifications, see Amazon S3 Event Notifications.

```
aws s3api get-bucket-notification-configuration --bucket amzn-s3-demo-bucket1
```

For more information and examples, see get-bucket-notification-configuration in the *AWS CLI Command Reference*.

The following returns the logging status for the bucket *amzn-s3-demo-bucket1*. For information about the bucket logging, see Logging requests with server access logging.

```
aws s3api get-bucket-logging --bucket amzn-s3-demo-bucket1
```

For more information and examples, see get-bucket-logging in the *AWS CLI Command Reference*.

Using the AWS SDKs

For examples of how to return general purpose bucket properties with the AWS SDKs, such as versioning, tags, and more, see Code examples in the *Amazon S3 API Reference*.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Listing Amazon S3 general purpose buckets

To return a list of general purpose buckets that you own, you can use ListBuckets. You can list your buckets by using the Amazon S3 console, the AWS Command Line Interface, or the AWS SDKs. For `ListBuckets` requests using the AWS CLI, AWS SDKs, and Amazon S3 REST API, AWS accounts that use the default service quota for buckets (10,000 buckets), support both paginated and unpaginated requests. Regardless of how many buckets you have in your account, you can create page sizes between 1 and 10,000 buckets to list all of your buckets. For paginated requests, `ListBuckets` requests return both the bucket names and the corresponding AWS Regions for each bucket. The following AWS Command Line Interface and AWS SDK examples show you how to use pagination in your `ListBuckets` request. Note that some AWS SDKs assist with pagination.

Permissions

To list all of your general purpose buckets, you must have the `s3:ListAllMyBuckets` permission. If you're encountering an `HTTP Access Denied (403 Forbidden)` error, see Troubleshoot access denied (403 Forbidden) errors in Amazon S3.

> **Important**
>
> We strongly recommend using only paginated `ListBuckets` requests. Unpaginated `ListBuckets` requests are only supported for AWS accounts set to the default general purpose bucket quota of 10,000. If you have an approved general purpose bucket quota above 10,000, you must send paginated `ListBuckets` requests to list your account's buckets. All unpaginated `ListBuckets` requests will be rejected for AWS accounts with a general purpose bucket quota greater than 10,000.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. On the **General purpose buckets** tab, you can see a list of your general purpose buckets.
4. To find buckets by name, enter a bucket name in the **Find buckets by name** field.

Using the AWS CLI

To use the AWS CLI to generate a listing of general purpose buckets, you can use the `ls` or `list-buckets` commands. The following examples show you how to create a paginated `list-buckets` request and an unpaginated `ls` request. To use these examples, replace the *`user input placeholders`*.

Example– List all the buckets in your account by using ls (unpaginated)

The following example command lists all the general purpose buckets in your account in a single non-paginated call. This call returns a list of all buckets in your account (up to 10,000 results):

```
$ aws s3 ls
```

For more information and examples, see List bucket and objects.

Example– List all the buckets in your account by using ls (paginated)

The following example command makes one or more paginated calls to list all the general purpose buckets in your account, returning 100 buckets per page:

```
$ aws s3 ls --page-size 100
```

For more information and examples, see List bucket and objects.

Example– List all the buckets in your account (paginated)

The following example provides a paginated `list-buckets` command to list all the general purpose buckets in your account. The `--max-items` and `--page-size` options limit the number of buckets listed to 100 per page.

```
$ aws s3api list-buckets /
    --max-items 100 /
    --page-size 100
```

If the number of items output (`--max-items`) is fewer than the total number of items returned by the underlying API calls, the output includes a continuation token, specified by the `starting-token` argument, that you can pass to a subsequent command to retrieve the next set of items. The following example shows how to use the `starting-token` value returned by the previous example. You can specify the `starting-code` to retrieve the next 100 buckets.

```
$ aws s3api list-buckets /
    --max-items 100 /
    --page-size 100 /
    --starting-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ==
```

Example– List all the buckets in an AWS Region (paginated)

The following example command uses the `--bucket-region` parameter to list up to 100 buckets in an account that are in the `us-east-2` Region. Requests made to a Regional endpoint that is different from the value specified in the `--bucket-region` parameter are not supported. For example, if you want to limit the response to your buckets in `us-east-2`, you must make your request to an endpoint in `us-east-2`.

```
$ aws s3api list-buckets /
    --region us-east-2 /
    --max-items 100 /
    --page-size 100 /
    --bucket-region us-east-2
```

Example– List all the buckets that begin with a specific bucket name prefix (paginated)

The following example command lists up to 100 buckets that have a name starting with the *amzn-s3-demo-bucket* prefix.

```
$ aws s3api list-buckets /
    --max-items 100 /
    --page-size 100 /
    --prefix amzn-s3-demo-bucket
```

Using the AWS SDKs

The following examples show you how to list your general purpose buckets by using the AWS SDKs

SDK for Python (Boto3)

Example– ListBuckets request (paginated)

```
import boto3

s3 = boto3.client('s3')
response = s3.list_buckets(MaxBuckets=100)
```

Example– ListBuckets response (paginated)

```
import boto3

s3 = boto3.client('s3')
response =
 s3.list_buckets(MaxBuckets=1,ContinuationToken="eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZVS
```

SDK for Java

For examples of how to list buckets with the AWS SDK for Java, see List buckets in the *Amazon S3 API Reference*.

SDK for Go

```
package main
import (
 "context"
 "fmt"
 "log"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/s3"
)
func main() {
 cfg, err := config.LoadDefaultConfig(context.TODO(), config.WithRegion("us-
east-2"))
 if err != nil {
  log.Fatal(err)
```

```
 }
 client := s3.NewFromConfig(cfg)
 maxBuckets := 1000
 resp, err := client.ListBuckets(context.TODO(), management
 portals3.ListBucketsInput{MaxBuckets: aws.Int32(int32(maxBuckets))})
 if err != nil {
  log.Fatal(err)
 }
 fmt.Println("S3 Buckets:")
 for _, bucket := range resp.Buckets {
     fmt.Println("- Name:", *bucket.Name)
     fmt.Println("-BucketRegion", *bucket.BucketRegion)
 }
 fmt.Println(resp.ContinuationToken == nil)
 fmt.Println(resp.Prefix == nil)
}
```

Emptying a general purpose bucket

You can empty a general purpose bucket's contents using the Amazon S3 console, AWS SDKs, or AWS Command Line Interface (AWS CLI). When you empty a general purpose bucket, you delete all the objects, but you keep the bucket. After you empty a bucket, it cannot be undone. Objects added to the bucket while the empty bucket action is in progress might be deleted. All objects (including all object versions and delete markers) in the bucket must be deleted before the bucket itself can be deleted.

When you empty a general purpose bucket that has S3 Versioning enabled or suspended, all versions of all the objects in the bucket are deleted. For more information, see Working with objects in a versioning-enabled bucket.

While emptying your bucket, we recommend that you also remove all incomplete multipart uploads. You can use multipart uploads to upload very large objects (up to 50 TB) as a set of parts for improved throughput and quicker recovery from network issues. In cases where the multipart upload process doesn't finish, the incomplete parts remain in the bucket (in an unusable state). These incomplete parts incur storage costs until the upload process is finished, or until the incomplete parts are removed. For more information, see Uploading and copying objects using multipart upload in Amazon S3.

As a best practice, we recommend configuring lifecycle rules to expire objects and incomplete multipart uploads that are older than a specific number of days. When you create your lifecycle

rule to expire incomplete multipart uploads, we recommend 7 days as a good starting point. For more information, see Setting an S3 Lifecycle configuration on a bucket.

Lifecycle expiration is an asynchronous process, so the rule might take some days to run before your bucket is empty. After the first time that Amazon S3 runs the rule, all objects that are eligible for expiration are marked for deletion. You're no longer charged for those objects that are marked for deletion. For more information, see How do I empty an Amazon S3 bucket using a lifecycle configuration rule?.

Using the S3 console

You can use the Amazon S3 console to empty a general purpose bucket, which deletes all of the objects in the bucket without deleting the bucket.

To empty an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the bucket list, select the option next to the name of the bucket that you want to empty, and then choose **Empty**.
4. On the **Empty bucket** page, confirm that you want to empty the bucket by entering the bucket name into the text field, and then choose **Empty**.
5. Monitor the progress of the bucket emptying process on the **Empty bucket: Status** page.

Using the AWS CLI

You can empty a general purpose bucket using the AWS CLI only if the bucket does not have Bucket Versioning enabled. If versioning is not enabled, you can use the `rm` (remove) AWS CLI command with the `--recursive` parameter to empty the bucket (or remove a subset of objects with a specific key name prefix).

The following `rm` command removes objects that have the key name prefix doc, for example, doc/doc1 and doc/doc2.

```
$ aws s3 rm s3://bucket-name/doc --recursive
```

Use the following command to remove all objects without specifying a prefix.

```
$ aws s3 rm s3://bucket-name --recursive
```

For more information, see Using high-level S3 commands with the AWS CLI in the *AWS Command Line Interface User Guide*.

> **Note**
>
> You can't remove objects from a bucket that has versioning enabled. Amazon S3 adds a delete marker when you delete an object, which is what this command does. For more information about S3 Bucket Versioning, see Retaining multiple versions of objects with S3 Versioning.

Using the AWS SDKs

You can use the AWS SDKs to empty a general purpose bucket or remove a subset of objects that have a specific key name prefix.

For an example of how to empty a bucket using AWS SDK for Java, see Deleting a general purpose bucket. The code deletes all objects, regardless of whether the bucket has versioning enabled, and then it deletes the bucket. To just empty the bucket, make sure that you remove the statement that deletes the bucket.

For more information about using other AWS SDKs, see Tools for Amazon Web Services.

Using a lifecycle configuration

To empty a large general purpose bucket, we recommend that you use an S3 Lifecycle configuration rule. Lifecycle expiration is an asynchronous process, so the rule might take some days to run before the bucket is empty. After the first time that Amazon S3 runs the rule, all objects that are eligible for expiration are marked for deletion. You're no longer charged for those objects that are marked for deletion. For more information, see How do I empty an Amazon S3 bucket using a lifecycle configuration rule?.

If you use a lifecycle configuration to empty your bucket, the configuration should include current versions, non-current versions, delete markers, and incomplete multipart uploads.

You can add lifecycle configuration rules to expire all objects or a subset of objects that have a specific key name prefix. For example, to remove all objects in a bucket, you can set a lifecycle rule to expire objects one day after creation.

Amazon S3 supports a bucket lifecycle rule that you can use to stop multipart uploads that don't complete within a specified number of days after being initiated. We recommend that you configure this lifecycle rule to minimize your storage costs. For more information, see Configuring a bucket lifecycle configuration to delete incomplete multipart uploads.

For more information about using a lifecycle configuration to empty a bucket, see Setting an S3 Lifecycle configuration on a bucket and Expiring objects.

Emptying a general purpose bucket with AWS CloudTrail configured

AWS CloudTrail tracks object-level data events in an Amazon S3 general purpose bucket, such as deleting objects. If you use a general purpose bucket as a destination to log your CloudTrail events and are deleting objects from that same bucket you may be creating new objects while emptying your bucket. To prevent this, stop your AWS CloudTrail trails. For more information about stopping your CloudTrail trails from logging events, see Turning off logging for a trail in the *AWS CloudTrail User Guide*.

Another alternative to stopping CloudTrail trails from being added to the bucket is to add a deny `s3:PutObject` statement to your bucket policy. If you want to store new objects in the bucket at a later time you will need to remove this deny `s3:PutObject` statement. For more information, see Object operations and IAM JSON policy elements: Effect in the *IAM User Guide*.

Deleting a general purpose bucket

You can delete an empty Amazon S3 general purpose bucket. For information about emptying a general purpose bucket, see the section called “Emptying a general purpose bucket”.

You can delete a bucket by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

> **Important**
>
> Before deleting a general purpose bucket, consider the following:
>
> - **If a bucket is deleted, it can't be restored by AWS.** Before deleting a bucket, make sure that you have backed up or replicated your data.
> - General purpose buckets names are unique within a global namespace. **If you delete a bucket in the shared global namespace, be aware that another AWS account can use the same general purpose bucket name for a new bucket and can therefore**

potentially receive requests intended for the deleted bucket. If you want to prevent this, or if you want to continue to use the same bucket name, don't delete the bucket. We recommend that you empty the bucket and keep it, and instead, block any bucket requests as needed. For buckets no longer in active use, we recommend emptying the bucket of all objects to minimize costs while retaining the bucket itself.

- We recommend creating buckets in your account regional namespace for assurance that only your account can ever own these bucket names. For more information, see Namespaces for general purpose buckets.
- When you delete a general purpose bucket, the bucket might not be instantly removed. Instead, Amazon S3 queues the bucket for deletion. Because Amazon S3 is distributed across AWS Regions, the deletion process takes time to fully propagate and achieve consistency throughout the system.
- If the bucket hosts a static website, and you created and configured an Amazon Route 53 hosted zone as described in Tutorial: Configuring a static website using a custom domain registered with Route 53, you must clean up the Route 53 hosted zone settings that are related to the bucket. For more information, see Step 2: Delete the Route 53 hosted zone.
- If the bucket receives log data from Elastic Load Balancing (ELB), we recommend that you stop the delivery of ELB logs to the bucket before deleting it. After you delete the bucket, if another user creates a bucket using the same name, your log data could potentially be delivered to that bucket. For information about ELB access logs, see Access logs for your Classic Load Balancer in the *User Guide for Classic Load Balancers* and Access logs for your Application Load Balancer in the *User Guide for Application Load Balancers*.

Troubleshooting

If you are unable to delete an Amazon S3 general purpose bucket, consider the following:

- **Make sure that the bucket is empty** – You can delete buckets only if they don't have any objects in them. Make sure that the bucket is empty. For information about emptying a bucket, see the section called “Emptying a general purpose bucket”.
- **Make sure that there aren't any access points attached** – You can delete buckets only if they don't have any S3 Access Points or Multi-Region Access Points attached within the same account. Before deleting the bucket, delete any same-account access points that are attached to the bucket.

- **Make sure that you have the `s3:DeleteBucket` permission** – If you can't delete a bucket, work with your IAM administrator to confirm that you have the `s3:DeleteBucket` permission. For information about how to view or update IAM permissions, see Changing permissions for an IAM user in the *IAM User Guide*. For troubleshooting information, see the section called "Troubleshoot access denied (403 Forbidden) errors".
- **Check for `s3:DeleteBucket` Deny statements in AWS Organizations service control policies (SCPs) and resource control policies (RCPs)** – SCPs and RCPs can deny the delete permission on a bucket. For more information, see service control policies and resource control policies in the *AWS Organizations User Guide*.
- **Check for `s3:DeleteBucket` Deny statements in your bucket policy** – If you have `s3:DeleteBucket` permissions in your IAM user or role policy and you can't delete a bucket, the bucket policy might include a `Deny` statement for `s3:DeleteBucket`. Buckets created by AWS Elastic Beanstalk have a policy containing this statement by default. Before you can delete the bucket, you must delete this statement or the bucket policy.

Prerequisites

Before you can delete a general purpose bucket, you must empty it. For information about emptying a bucket, see the section called "Emptying a general purpose bucket".

Using the S3 console

To delete an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, select the option button next to the name of the bucket that you want to delete, and then choose **Delete**.
4. On the **Delete bucket** page, confirm that you want to delete the bucket by entering the bucket name in the text field, and then choose **Delete bucket**.

> **Note**
>
> If the bucket contains any objects, empty the bucket before deleting it by choosing the **Empty bucket** button in the **This bucket is not empty** error alert and following the

instructions on the **Empty bucket** page. Then return to the **Delete bucket** page and delete the bucket.

5. To verify that you've deleted the bucket, open the **General purpose buckets** list and enter the name of the bucket that you deleted. If the bucket can't be found, your deletion was successful.

Using the AWS SDK for Java

To empty and delete a general purpose bucket using the AWS SDK for Java, you must first delete all objects in the general purpose bucket, and then delete the bucket.

For examples in other languages, see Use DeleteBucket with an AWS SDK or CLI in the *Amazon Simple Storage Service API Reference*. For information about using other AWS SDKs, see Tools for Amazon Web Services.

Java

To delete a bucket that contains objects using the AWS SDK for Java, you must delete all objects first, and then delete the bucket. This approach works for buckets with or without versioning enabled.

> **Note**
>
> For buckets without versioning enabled, you can delete all objects directly and then delete the bucket. For buckets with versioning enabled, you must delete all object versions before deleting the bucket.

For examples of how to delete a bucket with the AWS SDK for Java, see Delete a bucket in the *Amazon S3 API Reference*.

Using the AWS CLI

You can delete a general purpose bucket that contains objects with the AWS CLI if the bucket doesn't have versioning enabled. When you delete a bucket that contains objects, all the objects in the bucket are permanently deleted, including objects that have been transitioned to the S3 Glacier Flexible Retrieval storage class.

If your bucket doesn't have versioning enabled, you can use the `rb` (remove bucket) AWS CLI command with the `--force` parameter to delete the bucket and all the objects in it. This command deletes all the objects first and then deletes the bucket.

If versioning is enabled, using the `rb` command with the `--force` parameter doesn't delete versioned objects, so the bucket deletion fails because the bucket isn't empty. For more information about deleting versioned objects, see Deleting object versions.

To use the following command, replace *amzn-s3-demo-bucket* with the name of the bucket that you want to delete:

```
$ aws s3 rb s3://amzn-s3-demo-bucket --force
```

For more information, see Using High-Level S3 Commands with the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

Mount an Amazon S3 bucket as a local file system

Mountpoint for Amazon S3 is a high-throughput open source file client for mounting an Amazon S3 bucket as a local file system. With Mountpoint, your applications can access objects stored in Amazon S3 through file system operations, such as open and read. Mountpoint automatically translates these operations into S3 object API calls, giving your applications access to the elastic storage and throughput of Amazon S3 through a file interface.

Mountpoint for Amazon S3 is available for production use on your large-scale read-heavy applications: data lakes, machine learning training, image rendering, autonomous vehicle simulation, extract, transform, and load (ETL), and more.

Mountpoint supports basic file system operations, and can read files up to 50 TB in size. It can list and read existing files, and it can create new ones. It cannot modify existing files or delete directories, and it does not support symbolic links or file locking. Mountpoint is ideal for applications that do not need all of the features of a shared file system and POSIX-style permissions but require Amazon S3's elastic throughput to read and write large S3 datasets. For details, see Mountpoint file system behavior on GitHub. For workloads that require full POSIX support, we recommend Amazon FSx for Lustre and its support for linking Amazon S3 buckets.

Mountpoint for Amazon S3 is available only for Linux operating systems. You can use Mountpoint to access S3 objects in all storage classes except S3 Glacier Flexible Retrieval, S3 Glacier Deep

Archive, S3 Intelligent-Tiering Archive Access Tier, and S3 Intelligent-Tiering Deep Archive Access Tier.

Topics

- Installing Mountpoint
- Configuring and using Mountpoint
- Troubleshooting Mountpoint

Installing Mountpoint

You can download and install prebuilt packages of Mountpoint for Amazon S3 by using the command line. The instructions for downloading and installing Mountpoint vary, depending on which Linux operating system that you're using.

Topics

- Amazon Linux 2023 (AL2023)
- Other RPM-based distributions (Amazon Linux 2, Fedora, CentOS, RHEL)
- DEB-based distributions (Debian, Ubuntu)
- Other Linux distributions
- Verifying the signature of the Mountpoint for Amazon S3 package

Amazon Linux 2023 (AL2023)

Mountpoint is available directly in the Amazon Linux 2023 repository since AL2023 version 2023.9.20251110.

1. Install it by entering the following command:

```
sudo dnf install mount-s3
```

2. Verify that Mountpoint for Amazon S3 is successfully installed:

```
mount-s3 --version
```

   You should see output similar to the following:

```
mount-s3 1.21.0+1.amzn2023
```

Other RPM-based distributions (Amazon Linux 2, Fedora, CentOS, RHEL)

1. Copy the following download URL for your architecture.

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm
   ```

2. Download the Mountpoint for Amazon S3 package. Replace *download-link* with the appropriate download URL from the preceding step.

   ```
   wget download-link
   ```

3. (Optional) Verify the authenticity and integrity of the downloaded file. First, copy the appropriate signature URL for your architecture.

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm.asc
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.rpm.asc
   ```

   Next, see Verifying the signature of the Mountpoint for Amazon S3 package.

4. Install the package by using the following command:

   ```
   sudo yum install ./mount-s3.rpm
   ```

5. Verify that Mountpoint is successfully installed by entering the following command:

```
mount-s3 --version
```

You should see output similar to the following:

```
mount-s3 1.21.0
```

DEB-based distributions (Debian, Ubuntu)

1. Copy the download URL for your architecture.

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb
   ```

2. Download the Mountpoint for Amazon S3 package. Replace *download-link* with the appropriate download URL from the preceding step.

   ```
   wget download-link
   ```

3. (Optional) Verify the authenticity and integrity of the downloaded file. First, copy the signature URL for your architecture.

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.deb.asc
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.deb.asc
   ```

   Next, see Verifying the signature of the Mountpoint for Amazon S3 package.

4. Install the package by using the following command:

```
sudo apt-get install ./mount-s3.deb
```

5. Verify that Mountpoint for Amazon S3 is successfully installed by running the following command:

   ```
   mount-s3 --version
   ```

   You should see output similar to the following:

   ```
   mount-s3 1.21.0
   ```

Other Linux distributions

1. Consult your operating system documentation to install the FUSE and `libfuse2` packages, which are required.
2. Copy the download URL for your architecture.

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz
   ```

   ARM64 (Graviton):

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz
   ```

3. Download the Mountpoint for Amazon S3 package. Replace *`download-link`* with the appropriate download URL from the preceding step.

   ```
   wget download-link
   ```

4. (Optional) Verify the authenticity and integrity of the downloaded file. First, copy the signature URL for your architecture.

   x86_64:

   ```
   https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.tar.gz.asc
   ```

ARM64 (Graviton):

```
https://s3.amazonaws.com/mountpoint-s3-release/latest/arm64/mount-s3.tar.gz.asc
```

Next, see Verifying the signature of the Mountpoint for Amazon S3 package.

5. Install the package by using the following command:

```
sudo mkdir -p /opt/aws/mountpoint-s3 && sudo tar -C /opt/aws/mountpoint-s3 -xzf ./
mount-s3.tar.gz
```

6. Add the `mount-s3` binary to your `PATH` environment variable. In your `$HOME/.profile` file, append the following line:

```
export PATH=$PATH:/opt/aws/mountpoint-s3/bin
```

Save the `.profile` file, and run the following command:

```
source $HOME/.profile
```

7. Verify that Mountpoint for Amazon S3 is successfully installed by running the following command:

```
mount-s3 --version
```

You should see output similar to the following:

```
mount-s3 1.21.0
```

Verifying the signature of the Mountpoint for Amazon S3 package

1. Install GnuPG (the gpg command). It is required to verify the authenticity and integrity of a downloaded Mountpoint for Amazon S3 package. GnuPG is installed by default on Amazon Linux Amazon Machine Images (AMIs). After you installGnuPG, proceed to step 2.
2. Download the Mountpoint public key by running the following command:

```
wget https://s3.amazonaws.com/mountpoint-s3-release/public_keys/KEYS
```

3. Import the Mountpoint public key into your keyring by running the following command:

```
gpg --import KEYS
```

4. Verify the fingerprint of the Mountpoint public key by running the following command:

```
gpg --fingerprint mountpoint-s3@amazon.com
```

Confirm that the displayed fingerprint string matches one of the following:

```
8AEF E705 EBE3 29C0 948C  75A6 6F1C 3B3A EF4B 030B
673F E406 1506 BB46 9A0E  F857 BE39 7A52 B086 DA5A (older key)
```

If the fingerprint string doesn't match, do not finish installing Mountpoint, and contact AWS Support.

5. Download the package signature file. Replace *signature-link* with the appropriate signature link from the preceding sections.

```
wget signature-link
```

6. Verify the signature of the downloaded package by running the following command. Replace *signature-filename* with the file name from the previous step.

```
gpg --verify signature-filename
```

For example, on RPM-based distributions, including Amazon Linux, enter the following command:

```
gpg --verify mount-s3.rpm.asc
```

7. The output should include the phrase `Good signature`. If the output includes the phrase `BAD signature`, redownload the Mountpoint package file and repeat these steps. If the issue persists, do not finish installing Mountpoint, and contact AWS Support.

The output may include a warning about a trusted signature. This does not indicate a problem. It only means that you have not independently verified the Mountpoint public key.

Configuring and using Mountpoint

To use Mountpoint for Amazon S3, your host needs valid AWS credentials with access to the Amazon S3 bucket or buckets that you would like to mount. For different ways to authenticate, see Mountpoint AWS Credentials on GitHub.

For example, you can create a new AWS Identity and Access Management (IAM) user and role for this purpose. Make sure that this role has access to the bucket or buckets that you would like to mount. You can pass the IAM role to your Amazon EC2 instance with an instance profile.

Topics

- Using Mountpoint for Amazon S3
- Configuring caching in Mountpoint

Using Mountpoint for Amazon S3

Use Mountpoint for Amazon S3 to do the following:

1. Mount your Amazon S3 buckets.

   a. You can mount Amazon S3 buckets manually by using the `mount-s3` command.

      In the following example, replace *amzn-s3-demo-bucket* with the name of your S3 bucket, and replace *~/mnt* with the directory on your host where you want your S3 bucket to be mounted.

      ```
      mkdir ~/mnt
      mount-s3 amzn-s3-demo-bucket ~/mnt
      ```

      Because the Mountpoint client runs in the background by default, the *~/mnt* directory now gives you access to the objects in your Amazon S3 bucket.

   b. Alternatively, since Mountpoint v1.18, you can configure automatic mounting of Amazon S3 buckets when an instance starts up or reboots.

      For existing or running Amazon EC2 instances, find the `fstab` file in the `/etc/fstab` directory of your Linux system. Then, add a line to your `fstab` file. For example, to mount *amzn-s3-demo-bucket* using the prefix `example-prefix/` to your sytem path `/mnt/`

mountpoint, see the following. To use the following example, replace the *user input placeholders* with your own information.

```
s3://amzn-s3-demo-bucket/example-prefix/ /mnt/mountpoint mount-s3
 _netdev,nosuid,nodev,nofail,rw 0 0
```

See the following table for an explanation of the options used in the example.

Option	Description
_netdev	Specifies that the filesystem requires networking to mount.
nosuid	Specifies that the filesystem cannot contain set user ID files.
nodev	Specifies that the filesystem cannot contain special devices.
nofail	Specifies that failure to mount the filesystem should still allow the system to boot.
rw	Specifies that the mount point is created with read and write permissions. Alternati vely, use ro for read only.

For new Amazon EC2 instances, you can modify user data on an Amazon EC2 template and set up the fstab file as follows. To use the following example, replace the *user input placeholders* with your own information.

```
#!/bin/bash -e
MP_RPM=$(mktemp --suffix=.rpm)
curl https://s3.amazonaws.com/mountpoint-s3-release/latest/x86_64/mount-s3.rpm
 > $MP_RPM
yum install -y $MP_RPM
rm $MP_RPM
```

```
MNT_PATH=/mnt/mountpoint
echo "s3://amzn-s3-demo-bucket/ ${MNT_PATH} mount-s3
 _netdev,nosuid,nodev,rw,allow-other,nofail" >> /etc/fstab
mkdir $MNT_PATH

systemctl daemon-reload
mount -a
```

2. Access the objects in your Amazon S3 bucket through Mountpoint.

   After you mount your bucket locally, you can use common Linux commands, such as `cat` or `ls`, to work with your S3 objects. Mountpoint for Amazon S3 interprets keys in your Amazon S3 bucket as file system paths by splitting them on the forward slash (`/`) character. For example, if you have the object key `Data/2023-01-01.csv` in your bucket, you will have a directory named `Data` in your Mountpoint file system, with a file named `2023-01-01.csv` inside it.

   Mountpoint for Amazon S3 intentionally does not implement the full POSIX standard specification for file systems. Mountpoint is optimized for workloads that need high-throughput read and write access to data stored in Amazon S3 through a file system interface, but that otherwise do not rely on file system features. For more information, see Mountpoint for Amazon S3 file system behavior on GitHub. Customers that need richer file system semantics should consider other AWS file services, such as Amazon Elastic File System (Amazon EFS) or Amazon FSx.

3. Unmount your Amazon S3 bucket by using the `umount` command. This command unmounts your S3 bucket and exits Mountpoint.

   To use the following example command, replace *~/mnt* with the directory on your host where your S3 bucket is mounted.

   ```
   umount ~/mnt
   ```

   > **Note**
   >
   > To get a list of options for this command, run `umount --help`.

For additional Mountpoint configuration details, see Amazon S3 bucket configuration, and file system configuration on GitHub.

Configuring caching in Mountpoint

Mountpoint for Amazon S3 supports different types of data caching. To accelerate repeated read requests, you can opt in to the following:

- **Local cache** – You can use a local cache in your Amazon EC2 instance storage or an Amazon Elastic Block Store volume. If you repeatedly read the same data from the same compute instance and if you have unused space in your local instance storage for the repeatedly read dataset, you should opt in to a local cache.
- **Shared cache** – You can use a shared cache on S3 Express One Zone. If you repeatedly read small objects from multiple compute instances or if you do not know the size of your repeatedly read dataset and want to benefit from elasticity of cache size, you should opt in to the shared cache. Once you opt in, Mountpoint retains objects with sizes up to one megabyte in a directory bucket that uses S3 Express One Zone.
- **Combined local and shared cache** – If you have unused space in your local cache but also want a shared cache across multiple instances, you can opt in to both a local cache and shared cache.

Caching in Mountpoint is ideal for use cases where you repeatedly read the same data that doesn't change during the multiple reads. For example, you can use caching with machine learning training jobs that need to read a training dataset multiple times to improve model accuracy.

For more information about how to configure caching in Mountpoint, see the following examples.

Topics

- Local cache
- Shared cache
- Combined local and shared cache

Local cache

You can opt in to a local cache with the `--cache` *`CACHE_PATH`* flag. In the following example, replace *`CACHE_PATH`* with the filepath to the directory that you want to cache your data in. Replace *`amzn-s3-demo-bucket`* with the name of your Amazon S3 bucket, and replace *`~/mnt`* with the directory on your host where you want your S3 bucket to be mounted.

```
mkdir ~/mnt
mount-s3 --cache CACHE_PATH amzn-s3-demo-bucket ~/mnt
```

When you opt in to local caching while mounting an Amazon S3 bucket, Mountpoint creates an empty sub-directory at the configured cache location, if that sub-directory doesn't already exist. When you first mount a bucket and when you unmount, Mountpoint deletes the contents of the local cache.

> **Important**
>
> If you enable local caching, Mountpoint will persist unencrypted object content from your mounted Amazon S3 bucket at the local cache location provided at mount. In order to protect your data, you should restrict access to the data cache location by using file system access control mechanisms.

Shared cache

If you repeatedly read small objects (up to 1 MB) from multiple compute instances or the size of the dataset that you repeatedly read often exceeds the size of your local cache, you should use a shared cache in S3 Express One Zone. When you read the same data repeatedly from multiple instances, this improves latency by avoiding redundant requests to your mounted Amazon S3 bucket.

Once you opt in to the shared cache, you pay for the data cached in your directory bucket in S3 Express One Zone. You also pay for requests made against your data in the directory bucket in S3 Express One Zone. For more information, see Amazon S3 pricing. Mountpoint never deletes cached objects from directory buckets. To manage your storage costs, you should set up a Lifecycle policy on your directory bucket so that Amazon S3 expires the cached data in S3 Express One Zone after a period of time that you specify. For more information, see Mountpoint for Amazon S3 caching configuration on GitHub.

To opt in to caching in S3 Express One Zone when you mount an Amazon S3 bucket to your compute instance, use the `--cache-xz` flag and specify a directory bucket as your cache location. In the following example, replace the *`user input placeholders`*.

```
mount-s3 amzn-s3-demo-bucket ~/mnt --cache-xz amzn-s3-demo-bucket--usw2-az1--x-s3
```

Combined local and shared cache

If you have unused space on your instance but you also want to use a shared cache across multiple instances, you can opt in to both a local cache and shared cache. With this caching configuration,

you can avoid redundant read requests from the same instance to the shared cache in directory bucket when the required data is cached in local storage. This can reduce request costs and improve performance.

To opt in to both a local cache and shared cache when you mount an Amazon S3 bucket, you specify both cache locations by using the `--cache` and `--cache-xz` flags. To use the following example to opt into both a local and shared cache, replace the *`user input placeholders`*.

```
mount -s3 amzn-s3-demo-bucket ~/mnt --cache /path/to/mountpoint/cache --cache -xz amzn-s3-demo-bucket--usw2-az1--x-s3
```

For more information, Mountpoint for Amazon S3 caching configuration on GitHub.

> **Important**
>
> If you enable shared caching, Mountpoint will copy object content from your mounted Amazon S3 bucket into the S3 directory bucket that you provide as your shared cache location, making it accessible to any caller with access to the S3 directory bucket. To protect your cached data, you should follow the Security best practices for Amazon S3 to ensure that your buckets use the correct policies and are not publicly accessible. You should use a directory bucket dedicated to Mountpoint shared caching and grant access only to Mountpoint clients.

Troubleshooting Mountpoint

Mountpoint for Amazon S3 is backed by Support. If you need assistance, contact the AWS Support Center.

You can also review and submit Mountpoint Issues on GitHub.

If you discover a potential security issue in this project, we ask that you notify AWS Security through our vulnerability reporting page. Do not create a public GitHub issue.

If your application behaves unexpectedly with Mountpoint, you can inspect your log information to diagnose the problem.

Logging

By default, Mountpoint emits high-severity log information to syslog.

To view logs on most modern Linux distributions, including Amazon Linux, run the following `journald` command:

```
journalctl -e SYSLOG_IDENTIFIER=mount-s3
```

On other Linux systems, `syslog` entries are likely written to a file such as `/var/log/syslog`.

You can use these logs to troubleshoot your application. For example, if your application tries to overwrite an existing file, the operation fails, and you will see a line similar to the following in the log:

```
[WARN] open{req=12 ino=2}: mountpoint_s3::fuse: open failed: inode error: inode 2 (full
 key "README.md") is not writable
```

For more information, see Mountpoint for Amazon S3 Logging on GitHub.

Working with Storage Browser for Amazon S3

Storage Browser for S3 is an open source component that you can add to your web application to provide your end users with a simple graphical interface for data stored in Amazon S3. With Storage Browser for S3, you can provide authorized end users access to browse, download, upload, copy, and delete data in S3 directly from your own applications.

Storage Browser for S3 supports the following operations for files: `LIST`, `GET`, `PUT`, `COPY`, `UPLOAD`, and `DELETE`. To deliver high throughput data transfer, Storage Browser for S3 only displays the data that your end users are authorized to access and optimizes upload requests. Storage Browser also optimizes performance for faster load times, calculates checksums of the data that your end users upload, and accepts objects after confirming that your data integrity was maintained (in transit) over the public internet. You can control access to your data based on your end user's identity using AWS security and identity services, or your own managed services. You can also customize Storage Browser to match your existing application's design and branding.

Storage Browser for S3 is only available for web and intranet applications on the React framework. You can use Storage Browser to access Amazon S3 objects in all storage classes except S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, S3 Intelligent-Tiering Archive Access tier, and S3 Intelligent-Tiering Deep Archive Access tier.

Storage Browser for S3 is available to use with your web applications in the AWS Amplify React library. For more information about Storage Browser, see Storage Browser for S3.

Topics

- Using Storage Browser for S3
- Installing Storage Browser for S3
- Setting up Storage Browser for S3
- Configuring Storage Browser for S3
- Troubleshooting Storage Browser for S3

Using Storage Browser for S3

In Storage Browser for S3, a *location* is an S3 general purpose bucket or prefix, that you grant end users access to using S3 Access Grants, IAM policies, or your own managed authorization service. After you've authorized your end users to access a specific location, they can work with any data within that location.

The Storage Browser for S3 user interface has four main views:

- **Home page:** The home page lists the S3 locations that your users can access, as well as your permissions for each. This is the initial view for users that shows the root level S3 resources that your end users have access to and the permissions (READ/WRITE/READWRITE) for each S3 location.
- **Location details:** This view allows users to browse files and folders in S3, and upload or download files. (Note that in Storage Browser for S3, *objects* are known as files, and *prefixes* and *buckets* are known as folders.)
- **Location action:** After a user chooses an action (such as **Upload**) in Storage Browser, it opens up another view of the file location.
- **Vertical ellipsis:** The vertical ellipsis icon opens the drop-down list of actions.

When using Storage Browser for S3, be aware of the following limitations:

- Folders starting or ending with dots (.) aren’t supported.
- S3 Access Grants with WRITE only permission isn't supported.
- Storage Browser for S3 supports the PUT operation for files up to 160 GB in size.

- Storage Browser for S3 only supports the COPY operation for files smaller than 5 GB. If the file size exceeds 5 GB, Storage Browser fails the request.

Installing Storage Browser for S3

The fastest way to get started with Storage Browser is to clone one of the sample projects on GitHub. These sample projects can help you deploy production ready web apps for Storage Browser with preset integrations of AWS services for AWS Identity and Access Management so you can quickly connect authorized end users to data in S3.

For more information, see Quick start in the *Amplify Dev Center*.

Installing Storage Browser for S3 from GitHub

Alternatively, you can install Storage Browser for S3 from the latest version of `aws-amplify/ui-react-storage` and `aws-amplify` packages in the `aws-amplify` GitHub repository to start integrating Storage Browser into your existing application. When installing Storage Browser for S3, make sure to add the following dependencies to your `package.json` file:

```
"dependencies": {
    "aws-amplify/ui-react-storage": "latest",
    "aws-amplify": "latest",
  }
```

Alternatively, you can add the dependencies using Node Package Manager (NPM):

```
npm i --save @aws-amplify/ui-react-storage aws-amplify
```

Setting up Storage Browser for S3

To connect end users with Amazon S3 *locations*, you must first set up an authentication and authorization method. There are three methods to set up an authentication and authorization method with Storage Browser:

- Method 1: Managing data access for your customers and third party partners
- Method 2: Managing data access for your IAM principals for your AWS account
- Method 3: Managing data access at scale

Method 1: Managing data access for your customers and third party partners

With this method, you can use AWS Amplify Auth to manage access control and security for files. This method is ideal when you want to connect your customers or third party partners with data in S3. With this option, your customers can authenticate using social or enterprise identity providers.

You provide IAM credentials to your end users and third party partners using AWS Amplify Auth with an S3 bucket that's configured to use Amplify Storage. AWS Amplify Auth is built on Amazon Cognito, a fully managed customer identity and access management service where you can authenticate and authorize users from a built-in user directory or enterprise directory, or from consumer identity providers. The Amplify authorization model defines which prefixes the current authenticated user can access. For more information about how to set up authorization for AWS Amplify, see Set up storage.

To initialize the component with the Amplify authentication and storage methods, add the following code snippet to your web application:

```
import {
  createAmplifyAuthAdapter,
  createStorageBrowser,
} from '@aws-amplify/ui-react-storage/browser';
import "@aws-amplify/ui-react-storage/styles.css";

import config from './amplify_outputs.json';

Amplify.configure(config);

export const { StorageBrowser } = createStorageBrowser({
  config: createAmplifyAuthAdapter(),
});
```

Method 2: Managing data access for your IAM principals for your AWS account

If you want to manage access for your IAM principals or your AWS account directly, you can create an IAM role that has permissions to invoke the GetDataAccess S3 API operation. To set this up, you must create an S3 Access Grants instance to map out permissions for S3 general purpose buckets and prefixes to the specified IAM identities. The Storage Browser component (which must be called on the client side after obtaining the IAM credentials) will then invoke the ListCallerAccessGrants S3 API operation to fetch the available grants to the identity requester and populate the locations

in the component. After you obtain the `s3:GetDataAccess` permission, those credentials are then used by the Storage Browser component to request data access to S3.

```
import {
  createManagedAuthAdapter,
  createStorageBrowser,
} from '@aws-amplify/ui-react-storage/browser';
import "@aws-amplify/ui-react-storage/styles.css";

export const { StorageBrowser } = createStorageBrowser({
  config: createManagedAuthAdapter({
    credentialsProvider: async (options?: { forceRefresh?: boolean }) => {
      // return your credentials object
      return {
        credentials: {
          accessKeyId: 'my-access-key-id',
          secretAccessKey: 'my-secret-access-key',
          sessionToken: 'my-session-token',
          expiration: new Date()
        },
      }
    },
    // AWS `region` and `accountId`
    region: '',
    accountId: '',
    // call `onAuthStateChange` when end user auth state changes
    // to clear sensitive data from the `StorageBrowser` state
    registerAuthListener: (onAuthStateChange) => {},
  })
});
```

Method 3: Managing data access at scale

If you want to associate an S3 Access Grants instance to your IAM Identity Center for a more scalable solution (such as providing data access to your whole company), you can request data from Amazon S3 on behalf of the current authenticated user. For example, you can grant user groups in your corporate directory access to your data in S3. This approach allows you to centrally manage S3 Access Grants permissions for your users and groups, including the ones hosted on external providers such as Microsoft Entra, Okta, and others.

When using this method, the integration with the IAM Identity Center allows you to use existing user directories. Another benefit of an IAM Identity Center trusted identity propagation is that each

AWS CloudTrail data event for Amazon S3 contains a direct reference to the end user identity that accessed the S3 data.

If you have an application that supports OAuth 2.0 and your users need access from these applications to AWS services, we recommend that you use trusted identity propagation. With trusted identity propagation, a user can sign in to an application, and that application can pass the user's identity in any requests that access data in AWS services. This application interacts with IAM Identity Center on behalf of any authenticated users. For more information, see Using trusted identity propagation with customer managed applications.

Trusted identity propagation is an AWS IAM Identity Center feature that administrators of connected AWS services can use to grant and audit access to service data. Access to this data is based on user attributes such as group associations. Setting up trusted identity propagation requires collaboration between the administrators of connected AWS services and the IAM Identity Center administrators. For more information, see Prerequisites and considerations.

Setup

To set up Storage Browser authentication in the AWS Management Console using S3 Access Grants and IAM Identity Center trusted identity propagation, your applications must request data from Amazon S3 on behalf of the current authenticated user. With this approach, you can give users or groups of users from your corporate directory direct access to your S3 buckets, prefixes, or objects. This means that your application won't need to map any users to an IAM principal.

The following workflow outlines the steps for setting up Storage Browser for S3, using IAM Identity Center and S3 Access Grants:

Steps	Description
1	Enable IAM Identity Center for your AWS Organizations
2	Configure AWS Identity and Access Management Identity Center federation
3	Add a trusted token issuer in the AWS Identity and Access Management Identity Center console The trusted token issuer represents your external identity provider (IdP) within IAM Identity Center, enabling it to

Steps	Description
	recognize identity tokens for your application's authenticated users.
4	Create an IAM role for the `bootstrap_application` and `identity_bearer`
5	Create and configure your application in IAM Identity Center This application interacts with IAM Identity Center on behalf of authenticated users.
6	Add S3 Access Grants as a trusted application for identity propagation This step connects your application to S3 Access Grants, so that it can make requests to S3 Access Grants on behalf of authenticated users.
7	Create a grant to a user or group This step syncs users from AWS Identity and Access Management Identity Center with the System for Cross-dom ain Identity Management (SCIM). SCIM keeps your IAM Identity Center identities in sync with identities from your identity provider (IdP).
8	Create your Storage Browser for S3 component

Enable IAM Identity Center for your AWS Organizations

To enable IAM Identity Center for your AWS Organizations, perform the following steps:

1. Sign in to the AWS Management Console, using one of these methods:

    1. **New to AWS (root user) –** Sign in as the account owner by choosing Root user and entering your AWS account email address. On the next page, enter your password.
    2. **Already using AWS (IAM credentials) –** Sign in using your IAM credentials with administrative permissions.

2. Open the IAM Identity Center console.
3. Under **Enable IAM Identity Center**, choose **Enable**.

> **Note**
>
> IAM Identity Center requires the setup of AWS Organizations. If you haven't set up an organization, you can choose to have AWS create one for you. Choose **Create AWS organization** to complete this process.

4. Choose **Enable with AWS Organizations**.
5. Choose **Continue**.
6. (Optional) Add any tags that you want to associate with this organization instance.
7. (Optional) Configure the delegated administration.

> **Note**
>
> If you're using a multi-account environment, we recommend that you configure delegated administration. With delegated administration, you can limit the number of people who require access to the management account in AWS Organizations. For more information, see Delegated administration.

8. Choose **Save**.

AWS Organizations automatically sends a verification email to the address that is associated with your management account. There might be a delay before you receive the verification email. Make sure to verify your email address within 24 hours, before your verification email expires.

Configure AWS Identity and Access Management Identity Center federation

To use Storage Browser for S3 with corporate directory users, you must configure IAM Identity Center to use an external identity provider (IdP). You can use the preferred identity provider of your choice. However, be aware that each identity provider uses different configuration settings. For tutorials on using different external identity providers, see IAM Identity Center source tutorials.

> **Note**
>
> Make sure to record the issuer URL and the audience attributes of the identity provider that you've configured because you will need to refer to it in later steps. If you don't have

the required access or permissions to configure an IdP, you might need to contact the administrator of the external IdP to obtain them.

Add a trusted token issuer in the AWS Identity and Access Management Identity Center console

The trusted token issuer represents your external identity provider in the AWS Identity and Access Management Identity Center, and recognizes tokens for your application's authenticated users. The account owner of the IAM Identity Center instance in your AWS Organizations must perform these steps. These steps can be done either in the IAM Identity Center console, or programmatically.

To add a trusted token issuer in the AWS Identity and Access Management Identity Center console, perform the following steps:

1. Open the IAM Identity Center console.
2. Choose **Settings**.
3. Choose the **Authentication** tab.
4. Navigate to the **Trusted token issuers** section, and fill out the following details:
    1. Under **Issuer URL**, enter the URL of the external IdP that serves as the trusted token issuer. You might need to contact the administrator of the external IdP to obtain this information. For more information, see Using applications with a trusted token issuer.
    2. Under **Trusted token issuer name**, enter a name for the trusted token issuer. This name will appear in the list of trusted token issuers that you can select in *Step 8*, when an application resource is configured for identity propagation.
5. Update your **Map attributes** to your preferred application attribute, where each identity provider attribute is mapped to an IAM Identity Center attribute. For example, you might want to map the application attribute `email` to the IAM Identity Center user attribute `email`. To see the list of allowed user attributes in IAM Identity Center, see the table in Attribute mappings for AWS Managed Microsoft AD directory.
6. (Optional) If you want to add a resource tag, enter the key and value pair. To add multiple resource tags, choose **Add new tag** to generate a new entry and enter the key and value pairs.
7. Choose **Create trusted token issuer**.
8. After you finish creating the trusted token issuer, contact the application administrator to let them know the name of the trusted token issuer, so that they can confirm that the trusted token issuer is visible in the applicable console.

9. Make sure the application administrator selects this trusted token issuer in the applicable console to enable user access to the application from applications that are configured for trusted identity propagation.

Create an IAM role for the `bootstrap` application and `identity bearer`

To ensure that the `bootstrap` application and `identity bearer` users can properly work with each other, make sure to create two IAM roles. One IAM role is required for the `bootstrap` application and the other IAM role must be used for the identity bearer, or end users who are accessing the web application that requests access through S3 Access Grants. The `bootstrap` application receives the token issued by the identity provider and invokes the `CreateTokenWithIAM` API, exchanging this token with the one issued by the Identity Center.

Create an IAM role, such as `bootstrap-role`, with permissions such as the following. The following example IAM policy gives permissions to the `bootstrap-role` to perform the token exchange:

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Action": [
            "sso-oauth:CreateTokenWithIAM",
        ],
        "Resource": "arn:${Partition}:sso::${AccountId}:application/${InstanceId}/
${ApplicationId}",
        "Effect": "Allow"
    },
    {
        "Action": [
            "sts:AssumeRole",
            "sts:SetContext"
        ],
        "Resource": "arn:aws:iam::${AccountId}:role/identity-bearer-role",
        "Effect": "Allow"
    }]
}
```

Then, create a second IAM role (such as `identity-bearer-role`), which the identity broker uses to generate the IAM credentials. The IAM credentials returned from the identity broker to the web application are used by the Storage Browser for S3 component to allow access to S3 data:

```
{
    "Action": [
        "s3:GetDataAccess",
        "s3:ListCallerAccessGrants",
    ],
    "Resource": "arn:${Partition}:s3:${Region}:${Account}:access-grants/default",
    "Effect": "Allow"
}
```

This IAM role (`identity-bearer-role`) must use a trust policy with the following statement:

```
{
   "Effect": "Allow",
   "Principal": {
      "AWS": "arn:${Partition}:iam::${Account}:role/${RoleNameWithPath}"
   },
   "Action": [
       "sts:AssumeRole",
       "sts:SetContext"
   ]
}
```

Create and configure your application in IAM Identity Center

> **Note**
>
> Before you begin, make sure that you've created the required IAM roles from the previous step. You'll need to specify one of these IAM roles in this step.

To create and configure a customer managed application in AWS IAM Identity Center, perform the following steps:

1. Open the IAM Identity Center console.
2. Choose **Applications**.
3. Choose the **Customer managed** tab.
4. Choose **Add application**.
5. On the **Select application type** page, under **Setup preference**, choose **I have an application I want to set up**.

6. Under **Application type**, choose **OAuth 2.0**.
7. Choose **Next**. The **Specify application** page is displayed.
8. Under the **Application name and description**section, enter a **Display name** for the application, such as **`storage-browser-oauth`**.
9. Enter a **Description**. The application description appears in the IAM Identity Center console and API requests, but not in the AWS access portal.
10. Under **User and group assignment method**, choose **Do not require assignments**. This option allows all authorized IAM Identity Center users and groups access to this application.
11. Under **AWS access portal**, enter an application URL where users can access the application.
12. (Optional) If you want to add a resource tag, enter the key and value pair. To add multiple resource tags, choose **Add new tag** to generate a new entry and enter the key and value pairs.
13. Choose **Next**. The **Specify authentication page** displays.
14. Under **Authentication with trusted token issuer**, use the checkbox to select the trusted token issuer that you previously created.
15. Under **Configure selected trusted token issuers**, enter the aud claim. The **aud claim** identifies the audience of the JSON Web Token (JWT), and it is the name by which the trusted token issuer identifies this application.

    > **Note**
    >
    > You might need to contact the administrator of the external IdP to obtain this information.

16. Choose **Next**. The **Specify authentication credentials** page displays.
17. Under **Configuration method**, choose **Enter one or more IAM roles**.
18. Under **Enter IAM roles**, add the IAM role or Amazon Resource Name (ARN) for the identity bearer token. You must enter the IAM role that you created from the previous step for the identity broker application (for example, **`bootstrap-role`**).
19. Choose **Next**.
20. On the **Review and configure** page, review the details of your application configuration. If you need to modify any of the settings, choose **Edit** for the section that you want to edit and make your changes to.
21. Choose **Submit**. The details page of the application that you just added is displayed.

After you've set up your applications, your users can access your applications from within their AWS access portal based on the permission sets that you've created and the user access that you've assigned.

Add S3 Access Grants as a trusted application for identity propagation

After you set up your customer managed application, you must specify S3 Access Grants for identity propagation. S3 Access Grants vends credentials for users to access Amazon S3 data. When you sign in to your customer managed application, S3 Access Grants will pass your user identity to the trusted application.

Prerequisite: Make sure that you've set up S3 Access Grants (such as creating an S3 Access Grants instance and registering a location) before following these steps. For more information, see Getting started with S3 Access Grants.

To add S3 Access Grants for identity propagation to your customer managed application, perform the following steps:

1. Open the IAM Identity Center console.
2. Choose **Applications**.
3. Choose the **Customer managed** tab.
4. In the **Customer managed applications** list, select the OAuth 2.0 application for which you want to initiate access requests. This is the application that your users will sign in to.
5. On the **Details** page, under **Trusted applications for identity propagation**, choose **Specify trusted applications**.
6. Under Setup type, select Individual applications and specify access, and then choose **Next**.
7. On the **Select service** page, choose **S3 Access Grants**. S3 Access Grants has applications that you can use to define your own web application for trusted identity propagation.
8. Choose **Next**. You'll select your applications in the next step.
9. On the **Select applications** page, choose **Individual applications**, select the checkbox for each application that can receive requests for access, and then choose **Next**.
10. On the **Configure access** page, under **Configuration method**, choose either of the following:

    - **Select access per application –** Select this option to configure different access levels for each application. Choose the application for which you want to configure the access level, and then choose **Edit access**. In **Level of access to apply**, change the access levels as needed, and then choose **Save changes**.

- **Apply same level of access to all applications –** Select this option if you don't need to configure access levels on a per-application basis.

11. Choose **Next**.
12. On the **Review configuration** page, review the choices that you made.

> **Note**
>
> You'll want to make sure the `s3:access_grants:read_write` permission is granted for your users. This permission allows your users to retrieve credentials to access Amazon S3. Make sure to use either the IAM policy you created previously, or S3 Access Grants, to limit access to write operations.

13. To make changes, choose **Edit** for the configuration section that you want to make changes to. Then, make the required changes and choose **Save changes**.
14. Choose **Trust applications** to add the trusted application for identity propagation.

Create a grant to a user or group

In this step, you use IAM Identity Center to provision your users. You can use SCIM for automated or manual provisioning of users and groups. SCIM keeps your IAM Identity Center identities in sync with identities from your identity provider (IdP). This includes any provisioning, updates, and deprovisioning of users between your IdP and IAM Identity Center.

> **Note**
>
> This step is required because when S3 Access Grants is used with IAM Identity Center, local IAM Identity Center users aren't used. Instead, users must be synchronized from the identity provider with IAM Identity Center.

To synchronize users from your identity provider with IAM Identity Center, perform the following steps:

1. Enable automatic provisioning.
2. Generate an access token.

For examples of how to set up the identity provider with IAM Identity Center for your specific use case, see IAM Identity Center Identity source tutorials.

Create your Storage Browser for S3 component

After you've set up your IAM Identity Center instance and created grants in S3 Access Grants, open your React application. In the React application, use `createManagedAuthAdapter` to set up the authorization rules. You must provide a credentials provider to return the credentials you acquired from IAM Identity Center. You can then call `createStorageBrowser` to initialize the Storage Browser for S3 component:

```
import {
    createManagedAuthAdapter,
    createStorageBrowser,
} from '@aws-amplify/ui-react-storage/browser';
import '@aws-amplify/ui-react-storage/styles.css';

export const { StorageBrowser } = createStorageBrowser({
   config: createManagedAuthAdapter({
    credentialsProvider: async (options?: { forceRefresh?: boolean }) => {
      // return your credentials object
      return {
        credentials: {
          accessKeyId: 'my-access-key-id',
          secretAccessKey: 'my-secret-access-key',
          sessionToken: 'my-session-token',
          expiration: new Date(),
        },
      }
    },
    // AWS `region` and `accountId` of the S3 Access Grants Instance.
    region: '',
    accountId: '',
    // call `onAuthStateChange` when end user auth state changes
    // to clear sensitive data from the `StorageBrowser` state
    registerAuthListener: (onAuthStateChange) => {},
  })
});
```

Then, create a mechanism to exchange the JSON web tokens (JWTs) from your web application with the IAM credentials from IAM Identity Center. For more information about how to exchange the JWT, see the following resources:

- How to develop a user-facing data application with IAM Identity Center and S3 Access Grants post in *AWS Storage Blog*
- Scaling data access with S3 Access Grants post in *AWS Storage Blog*
- S3 Access Grants workshop on *AWS workshop studio*
- S3 Access Grants workshop on *GitHub*

Then, set up an API endpoint to handle requests for fetching credentials. To validate the JSON web token (JWT) exchange, perform the following steps:

1. Retrieve the JSON web token from the authorization header for incoming requests.
2. Validate the token using the public keys from the specified JSON web key set (JWKS) URL.
3. Verify the token's expiration, issuer, subject, and audience claims.

To exchange the identity provider's JSON web token with AWS IAM credentials, perform the following steps:

> **Tip**
>
> Make sure to avoid logging any sensitive information. We recommend that you use error handling controls for missing authorization, expired tokens, and other exceptions. For more information, see the Implementing AWS Lambda error handling patterns post in *AWS Compute Blog*.

1. Verify that the required **Permission** and **Scope** parameters are provided in the request.
2. Use the CreateTokenWithIAM API to exchange the JSON web token for an IAM Identity Center token.

   > **Note**
   >
   > After the IdP JSON web token is used, it can't be used again. A new token must be used to exchange with IAM Identity Center.

3. Use the AssumeRole API operation to assume a transient role using the IAM Identity Center token. Make sure to use the identity bearer role, also known as the role that carries the identity context (for example, **`identity-bearer-role`**) to request the credentials.
4. Return the IAM credentials to the web application.

> **Note**
>
> Make sure that you've set up a proper logging mechanism. Responses are returned in a standardized JSON format with an appropriate HTTP status code.

Configuring Storage Browser for S3

To allow Storage Browser for S3 access to S3 buckets, the Storage Browser component makes the REST API calls to Amazon S3. By default, cross-origin resource sharing (CORS) isn't enabled on S3 buckets. As a result, you must enable CORS for each S3 bucket that Storage Browser is accessing data from.

For example, to enable CORS on your S3 bucket, you can update your CORS policy like this:

```
[
    {
        "ID": "S3CORSRuleId1",
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "GET",
            "HEAD",
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": [
            "last-modified",
            "content-type",
            "content-length",
            "etag",
```

```
            "x-amz-version-id",
            "x-amz-request-id",
            "x-amz-id-2",
            "x-amz-cf-id",
            "x-amz-storage-class",
            "date",
            "access-control-expose-headers"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

Troubleshooting Storage Browser for S3

If you're experiencing issues with Storage Browser for S3, make sure to review the following troubleshooting tips:

- Avoid trying to use the same token (`idToken` or `accessToken`) for multiple requests. Tokens can't be reused. This will result in a request failure.
- Make sure that the IAM credentials that you provide to the Storage Browser component includes permissions to invoke the `s3:GetDataAccess` operation. Otherwise, your end users won't be able to access your data.

Alternatively, you can check the following resources:

- Storage Browser for S3 is backed by AWS Support. If you need assistance, contact the AWS Support Center.
- If you're having trouble with Storage Browser for S3 or would like to submit feedback, visit the Amplify GitHub page.
- If you discover a potential security issue in this project, you can notify AWS Security through the AWS Vulnerability Reporting page.

Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration

Amazon S3 Transfer Acceleration is a bucket-level feature that enables fast, easy, and secure transfers of files over long distances between your client and an S3 general purpose bucket.

Transfer Acceleration is designed to optimize transfer speeds from across the world into S3 general purpose buckets. Transfer Acceleration takes advantage of the globally distributed edge locations in Amazon CloudFront. As the data arrives at an edge location, the data is routed to Amazon S3 over an optimized network path.

When you use Transfer Acceleration, additional data transfer charges might apply. For more information about pricing, see Amazon S3 pricing.

Why use Transfer Acceleration?

You might want to use Transfer Acceleration on a general purpose bucket for various reasons:

- Your customers upload to a centralized general purpose bucket from all over the world.
- You transfer gigabytes to terabytes of data on a regular basis across continents.
- You can't use all of your available bandwidth over the internet when uploading to Amazon S3.

For more information about when to use Transfer Acceleration, see Amazon S3 FAQs.

Requirements for using Transfer Acceleration

The following are required when you are using Transfer Acceleration on an S3 bucket:

- Transfer Acceleration is only supported on virtual-hosted style requests. For more information about virtual-hosted style requests, see Making requests using the REST API in the *Amazon S3 API Reference*.
- The name of the bucket used for Transfer Acceleration must be DNS-compliant and must not contain periods (".").
- Transfer Acceleration must be enabled on the bucket. For more information, see Enabling and using S3 Transfer Acceleration.

  After you enable Transfer Acceleration on a bucket, it might take up to 20 minutes before the data transfer speed to the bucket increases.

> **Note**
>
> Transfer Acceleration is currently supported for buckets located in the following Regions:
>
> - Asia Pacific (Tokyo) (ap-northeast-1)

- Asia Pacific (Seoul) (ap-northeast-2)
- Asia Pacific (Mumbai) (ap-south-1)
- Asia Pacific (Singapore) (ap-southeast-1)
- Asia Pacific (Sydney) (ap-southeast-2)
- Canada (Central) (ca-central-1)
- Europe (Frankfurt) (eu-central-1)
- Europe (Ireland) (eu-west-1)
- Europe (London) (eu-west-2)
- Europe (Paris) (eu-west-3)
- South America (São Paulo) (sa-east-1)
- US East (N. Virginia) (us-east-1)
- US East (Ohio) (us-east-2)
- US West (N. California) (us-west-1)
- US West (Oregon) (us-west-2)

- To access the bucket that is enabled for Transfer Acceleration, you must use the endpoint *bucket-name*`.s3-accelerate.amazonaws.com`. Or, use the dual-stack endpoint *bucket-name*`.s3-accelerate.dualstack.amazonaws.com` to connect to the enabled bucket over IPv6. You can continue to use the regular endpoints for standard data transfer.
- You must be the bucket owner to set the transfer acceleration state. The bucket owner can assign permissions to other users to allow them to set the acceleration state on a bucket. The `s3:PutAccelerateConfiguration` permission permits users to enable or disable Transfer Acceleration on a bucket. The `s3:GetAccelerateConfiguration` permission permits users to return the Transfer Acceleration state of a bucket, which is either `Enabled` or `Suspended`.

The following sections describe how to get started and use Amazon S3 Transfer Acceleration for transferring data.

Topics

- Getting started with Amazon S3 Transfer Acceleration
- Enabling and using S3 Transfer Acceleration
- Using the Amazon S3 Transfer Acceleration Speed Comparison tool

Getting started with Amazon S3 Transfer Acceleration

You can use Amazon S3 Transfer Acceleration for fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. Transfer Acceleration uses the globally distributed edge locations in Amazon CloudFront. As the data arrives at an edge location, data is routed to Amazon S3 over an optimized network path.

To get started using Amazon S3 Transfer Acceleration, perform the following steps:

1. **Enable Transfer Acceleration on a bucket**

   You can enable Transfer Acceleration on a bucket any of the following ways:

   - Use the Amazon S3 console.
   - Use the REST API PutBucketAccelerateConfiguration operation.
   - Use the AWS CLI and AWS SDKs. For more information, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

   For more information, see Enabling and using S3 Transfer Acceleration.

   > **Note**
   >
   > For your bucket to work with transfer acceleration, the bucket name must conform to DNS naming requirements and must not contain periods (`.`).

2. **Transfer data to and from the acceleration-enabled bucket**

   Use one of the following `s3-accelerate` endpoint domain names:

   - To access an acceleration-enabled bucket, use *`bucket-name`*`.s3-accelerate.amazonaws.com`.
   - To access an acceleration-enabled bucket over IPv6, use *`bucket-name`*`.s3-accelerate.dualstack.amazonaws.com`.

     Amazon S3 dual-stack endpoints support requests to S3 buckets over IPv6 and IPv4. The Transfer Acceleration dual-stack endpoint only uses the virtual hosted-style type of endpoint name. For more information, see Making requests to Amazon S3 over IPv6 in the *Amazon S3 API Reference* and Using Amazon S3 dual-stack endpoints in the *Amazon S3 API Reference*.

> **Note**
>
> Your data transfer application must use one of the following two types of endpoints to access the bucket for faster data transfer: `.s3-accelerate.amazonaws.com` or `.s3-accelerate.dualstack.amazonaws.com` for the dual-stack endpoint. If you want to use standard data transfer, you can continue to use the regular endpoints.

You can point your Amazon S3 PUT object and GET object requests to the `s3-accelerate` endpoint domain name after you enable Transfer Acceleration. For example, suppose that you currently have a REST API application using PutObject that uses the hostname *amzn-s3-demo-bucket*`.s3.us-east-1.amazonaws.com` in the PUT request. To accelerate the PUT, you change the hostname in your request to *amzn-s3-demo-bucket*`.s3-accelerate.amazonaws.com`. To go back to using the standard upload speed, change the name back to *amzn-s3-demo-bucket*`.s3.us-east-1.amazonaws.com`.

After Transfer Acceleration is enabled, it can take up to 20 minutes for you to realize the performance benefit. However, the accelerate endpoint is available as soon as you enable Transfer Acceleration.

You can use the accelerate endpoint in the AWS CLI, AWS SDKs, and other tools that transfer data to and from Amazon S3. If you are using the AWS SDKs, some of the supported languages use an accelerate endpoint client configuration flag so you don't need to explicitly set the endpoint for Transfer Acceleration to *bucket-name*`.s3-accelerate.amazonaws.com`. For examples of how to use an accelerate endpoint client configuration flag, see Enabling and using S3 Transfer Acceleration.

You can use all Amazon S3 operations through the transfer acceleration endpoints *except* for the following:

- ListBuckets
- CreateBucket
- DeleteBucket

Also, Amazon S3 Transfer Acceleration does not support cross-Region copies using CopyObject.

Enabling and using S3 Transfer Acceleration

You can use Amazon S3 Transfer Acceleration to transfer files quickly and securely over long distances between your client and an S3 general purpose bucket. You can enable Transfer Acceleration using the S3 console, the AWS Command Line Interface (AWS CLI), API, or the AWS SDKs.

This section provides examples of how to enable Amazon S3 Transfer Acceleration on a bucket and use the acceleration endpoint for the enabled bucket.

For more information about Transfer Acceleration requirements, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

Using the S3 console

> **Note**
>
> If you want to compare accelerated and non-accelerated upload speeds, open the Amazon S3 Transfer Acceleration Speed Comparison tool.
> The Speed Comparison tool uses multipart upload to transfer a file from your browser to various AWS Regions with and without Amazon S3 transfer acceleration. You can compare the upload speed for direct uploads and transfer accelerated uploads by Region.

To enable transfer acceleration for an S3 general purpose bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the **General purpose buckets** list, choose the name of the bucket that you want to enable transfer acceleration for.
4. Choose **Properties**.
5. Under **Transfer acceleration**, choose **Edit**.
6. Choose **Enable**, and choose **Save changes**.

To access accelerated data transfers

1. After Amazon S3 enables transfer acceleration for your bucket, view the **Properties** tab for the bucket.
2. Under **Transfer acceleration**, **Accelerated endpoint** displays the transfer acceleration endpoint for your bucket. Use this endpoint to access accelerated data transfers to and from your bucket.

   If you suspend transfer acceleration, the accelerate endpoint no longer works.

Using the AWS CLI

The following are examples of AWS CLI commands used for Transfer Acceleration. For instructions on setting up the AWS CLI, see Developing with Amazon S3 using the AWS CLI in the *Amazon S3 API Reference*.

Enabling Transfer Acceleration on a bucket

Use the AWS CLI put-bucket-accelerate-configuration command to enable or suspend Transfer Acceleration on a bucket.

The following example sets `Status=Enabled` to enable Transfer Acceleration on a bucket named *`amzn-s3-demo-bucket`*. To suspend Transfer Acceleration, use `Status=Suspended`.

Example

```
$ aws s3api put-bucket-accelerate-configuration --bucket amzn-s3-demo-bucket --
accelerate-configuration Status=Enabled
```

Using Transfer Acceleration

You can direct all Amazon S3 requests made by `s3` and `s3api` AWS CLI commands to the accelerate endpoint: `s3-accelerate.amazonaws.com`. To do this, set the configuration value `use_accelerate_endpoint` to `true` in a profile in your AWS Config file. Transfer Acceleration must be enabled on your bucket to use the accelerate endpoint.

All requests are sent using the virtual style of bucket addressing: *`amzn-s3-demo-bucket`*`.s3-accelerate.amazonaws.com`. Any `ListBuckets`, `CreateBucket`, and `DeleteBucket` requests are not sent to the accelerate endpoint because the endpoint doesn't support those operations.

For more information about `use_accelerate_endpoint`, see AWS CLI S3 Configuration in the *AWS CLI Command Reference*.

The following example sets `use_accelerate_endpoint` to `true` in the default profile.

Example

```
$ aws configure set default.s3.use_accelerate_endpoint true
```

If you want to use the accelerate endpoint for some AWS CLI commands but not others, you can use either one of the following two methods:

- Use the accelerate endpoint for any `s3` or `s3api` command by setting the `--endpoint-url` parameter to `https://s3-accelerate.amazonaws.com`.
- Set up separate profiles in your AWS Config file. For example, create one profile that sets `use_accelerate_endpoint` to `true` and a profile that does not set `use_accelerate_endpoint`. When you run a command, specify which profile you want to use, depending upon whether you want to use the accelerate endpoint.

Uploading an object to a bucket enabled for Transfer Acceleration

The following example uploads a file to a bucket named *amzn-s3-demo-bucket* that's been enabled for Transfer Acceleration by using the default profile that has been configured to use the accelerate endpoint.

Example

```
$ aws s3 cp file.txt s3://amzn-s3-demo-bucket/key-name --region region
```

The following example uploads a file to a bucket enabled for Transfer Acceleration by using the `--endpoint-url` parameter to specify the accelerate endpoint.

Example

```
$ aws configure set s3.addressing_style virtual
$ aws s3 cp file.txt s3://amzn-s3-demo-bucket/key-name --region region --endpoint-url
 https://s3-accelerate.amazonaws.com
```

Using the AWS SDKs

The following are examples of using Transfer Acceleration to upload objects to Amazon S3 using the AWS SDK. Some of the AWS SDK supported languages (for example, Java and .NET) use an accelerate endpoint client configuration flag so you don't need to explicitly set the endpoint for Transfer Acceleration to *bucket-name*.s3-accelerate.amazonaws.com.

Java

To use an accelerate endpoint to upload an object to Amazon S3 with the AWS SDK for Java, you can:

- Create an S3Client that is configured to use accelerate endpoints. All buckets that the client accesses must have Transfer Acceleration enabled.
- Enable Transfer Acceleration on a specified bucket. This step is necessary only if the bucket you specify doesn't already have Transfer Acceleration enabled.
- Verify that transfer acceleration is enabled for the specified bucket.
- Upload a new object to the specified bucket using the bucket's accelerate endpoint.

For more information about using Transfer Acceleration, see Getting started with Amazon S3 Transfer Acceleration.

The following code example shows how to configure Transfer Acceleration with the AWS SDK for Java.

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketAccelerateStatus;
import
 software.amazon.awssdk.services.s3.model.GetBucketAccelerateConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.PutBucketAccelerateConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.AccelerateConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.core.exception.SdkClientException;

public class TransferAcceleration {
```

```
   public static void main(String[] args) {
       Region clientRegion = Region.US_EAST_1;
       String bucketName = "*** Provide bucket name ***";
       String keyName = "*** Provide key name ***";

       try {
           // Create an Amazon S3 client that is configured to use the accelerate
endpoint.
           S3Client s3Client = S3Client.builder()
                   .region(clientRegion)
                   .credentialsProvider(ProfileCredentialsProvider.create())
                   .accelerate(true)
                   .build();

           // Enable Transfer Acceleration for the specified bucket.
           s3Client.putBucketAccelerateConfiguration(
                   PutBucketAccelerateConfigurationRequest.builder()
                           .bucket(bucketName)

.accelerateConfiguration(AccelerateConfiguration.builder()
                                   .status(BucketAccelerateStatus.ENABLED)
                                   .build())
                           .build());

           // Verify that transfer acceleration is enabled for the bucket.
           String accelerateStatus = s3Client.getBucketAccelerateConfiguration(
                   GetBucketAccelerateConfigurationRequest.builder()
                           .bucket(bucketName)
                           .build())
                   .status().toString();
           System.out.println("Bucket accelerate status: " + accelerateStatus);

           // Upload a new object using the accelerate endpoint.
           s3Client.putObject(PutObjectRequest.builder()
                           .bucket(bucketName)
                           .key(keyName)
                           .build(),
                   RequestBody.fromString("Test object for transfer
acceleration"));
           System.out.println("Object \"" + keyName + "\" uploaded with transfer
acceleration.");
       } catch (S3Exception e) {
           // The call was transmitted successfully, but Amazon S3 couldn't process
           // it, so it returned an error response.
```

```
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

The following example shows how to use the AWS SDK for .NET to enable Transfer Acceleration on a bucket. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TransferAccelerationTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            EnableAccelerationAsync().Wait();
        }

        static async Task EnableAccelerationAsync()
        {
                try
                {
                    var putRequest = new PutBucketAccelerateConfigurationRequest
```

```
                    {
                        BucketName = bucketName,
                        AccelerateConfiguration = new AccelerateConfiguration
                        {
                            Status = BucketAccelerateStatus.Enabled
                        }
                    };
                    await
 s3Client.PutBucketAccelerateConfigurationAsync(putRequest);

                    var getRequest = new GetBucketAccelerateConfigurationRequest
                    {
                        BucketName = bucketName
                    };
                    var response = await
 s3Client.GetBucketAccelerateConfigurationAsync(getRequest);

                    Console.WriteLine("Acceleration state = '{0}' ",
 response.Status);
                }
                catch (AmazonS3Exception amazonS3Exception)
                {
                    Console.WriteLine(
                        "Error occurred. Message:'{0}' when setting transfer
 acceleration",
                        amazonS3Exception.Message);
                }
        }
    }
}
```

When uploading an object to a bucket that has Transfer Acceleration enabled, you specify using the acceleration endpoint at the time of creating a client.

```
var client = new AmazonS3Client(new AmazonS3Config
            {
                RegionEndpoint = TestRegionEndpoint,
                UseAccelerateEndpoint = true
            }
```

JavaScript

For an example of enabling Transfer Acceleration by using the AWS SDK for JavaScript, see PutBucketAccelerateConfiguration command in the *AWS SDK for JavaScript API Reference*.

Python (Boto)

For an example of enabling Transfer Acceleration by using the AWS SDK for Python (Boto3), see put_bucket_accelerate_configuration in the *AWS SDK for Python (Boto3) API Reference*.

Other

For information about using other AWS SDKs, see Sample Code and Libraries.

Using the REST API

Use the REST API `PutBucketAccelerateConfiguration` operation to enable accelerate configuration on an existing bucket.

For more information, see PutBucketAccelerateConfiguration in the *Amazon Simple Storage Service API Reference*.

Using the Amazon S3 Transfer Acceleration Speed Comparison tool

You can use the Amazon S3 Transfer Acceleration Speed Comparison tool to compare accelerated and non-accelerated upload speeds across Amazon S3 Regions. The Speed Comparison tool uses multipart uploads to transfer a file from your browser to various Amazon S3 Regions with and without using Transfer Acceleration.

You can access the Speed Comparison tool by using either of the following methods:

- Copy the following URL into your browser window, replacing *`region`* with the AWS Region that you are using (for example, `us-west-2`) and *`amzn-s3-demo-bucket`* with the name of the bucket that you want to evaluate:

  `https://s3-accelerate-speedtest.s3-accelerate.amazonaws.com/en/accelerate-speed-comparsion.html?region=`*`region`*`&origBucketName=`*`amzn-s3-demo-bucket`*

  For a list of the Regions supported by Amazon S3, see Amazon S3 endpoints and quotas in the *AWS General Reference*.
- Use the Amazon S3 console.

Using Requester Pays general purpose buckets for storage transfers and usage

In general, bucket owners pay for all Amazon S3 storage and data transfer costs that are associated with their bucket. However, you can configure a general purpose bucket to be a *Requester Pays* bucket. With Requester Pays buckets, the requester instead of the bucket owner pays the cost of the request and the data download from the bucket. The bucket owner always pays the cost of storing data.

Typically, you configure buckets to be Requester Pays buckets when you want to share data but not incur charges associated with others accessing the data. For example, you might use Requester Pays buckets when making available large datasets, such as zip code directories, reference data, geospatial information, or web crawling data.

> **Important**
>
> If you enable Requester Pays on a general purpose bucket, anonymous access to that bucket is not allowed.

You must authenticate all requests involving Requester Pays buckets. The request authentication enables Amazon S3 to identify and charge the requester for their use of the Requester Pays bucket.

When the requester assumes an AWS Identity and Access Management (IAM) role before making their request, the account to which the role belongs is charged for the request. For more information about IAM roles, see IAM roles in the *IAM User Guide*.

After you configure a bucket to be a Requester Pays bucket, requesters must show they understand that they will be charged for the request and for the data download. To show they accept the charges, requesters must either include `x-amz-request-payer` as a header in their API request for DELETE, GET, HEAD, POST, and PUT requests, or add the `RequestPayer` parameter in their REST request. For CLI requests, requesters can use the `--request-payer` parameter.

Example– Using Requester Pays when deleting an object

To use the following DeleteObjectVersion API example, replace the *`user input placeholders`* with your own information.

```
DELETE /Key+?versionId=VersionId HTTP/1.1
Host: Bucket.s3.amazonaws.com
x-amz-mfa: MFA
x-amz-request-payer: RequestPayer
x-amz-bypass-governance-retention: BypassGovernanceRetention
x-amz-expected-bucket-owner: ExpectedBucketOwner
```

If the requester restores objects by using the RestoreObject API, Requester Pays is supported as long as the `x-amz-request-payer` header or the `RequestPayer` parameter are in the request; however, the requester only pays for the cost of the request. The bucket owner pays the retrieval charges.

Requester Pays buckets do not support the following:

- Anonymous requests
- SOAP requests
- Using a Requester Pays bucket as the target bucket for end-user logging, or vice versa. However, you can turn on end-user logging on a Requester Pays bucket where the target bucket is not a Requester Pays bucket.

How Requester Pays charges work

The charge for successful Requester Pays requests is straightforward: The requester pays for the data transfer and the request, and the bucket owner pays for the data storage. However, the bucket owner is charged for the request under the following conditions:

- The request returns an `AccessDenied` (HTTP `403 Forbidden`) error and the request is initiated inside the bucket owner's individual AWS account or AWS organization.
- The request is a SOAP request.

For more information about Requester Pays, see the following topics.

Topics

- Configuring Requester Pays on a bucket
- Retrieving the requestPayment configuration using the REST API
- Downloading objects from Requester Pays buckets

Configuring Requester Pays on a bucket

You can configure an Amazon S3 bucket to be a *Requester Pays* bucket so that the requester pays the cost of the request and data download instead of the bucket owner.

This section provides examples of how to configure Requester Pays on an Amazon S3 bucket using the console and the REST API.

Using the S3 console

To enable Requester Pays for an S3 general purpose bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the **General purpose buckets** list, choose the name of the bucket that you want to enable Requester Pays for.
4. Choose **Properties**.
5. Under **Requester pays**, choose **Edit**.
6. Choose **Enable**, and choose **Save changes**.

   Amazon S3 enables Requester Pays for your bucket and displays your **Bucket overview**. Under **Requester pays**, you see **Enabled**.

Using the REST API

Only the bucket owner can set the `RequestPaymentConfiguration.payer` configuration value of a bucket to `BucketOwner` (the default) or `Requester`. Setting the `requestPayment` resource is optional. By default, the bucket is not a Requester Pays bucket.

To revert a Requester Pays bucket to a regular bucket, you use the value `BucketOwner`. Typically, you would use `BucketOwner` when uploading data to the Amazon S3 bucket, and then you would set the value to `Requester` before publishing the objects in the bucket.

To set requestPayment

- Use a PUT request to set the `Payer` value to `Requester` on a specified bucket.

```
PUT ?requestPayment HTTP/1.1
```

```
Host: [BucketName].s3.amazonaws.com
Content-Length: 173
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]

<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

If the request succeeds, Amazon S3 returns a response similar to the following.

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Length: 0
Connection: close
Server: AmazonS3
x-amz-request-charged:requester
```

You can set Requester Pays only at the bucket level. You can't set Requester Pays for specific objects within the bucket.

You can configure a bucket to be `BucketOwner` or `Requester` at any time. However, there might be a few minutes before the new configuration value takes effect.

> **Note**
>
> Bucket owners who give out presigned URLs should consider carefully before configuring a bucket to be Requester Pays, especially if the URL has a long lifetime. The bucket owner is charged each time the requester uses a presigned URL that uses the bucket owner's credentials.

Retrieving the requestPayment configuration using the REST API

You can determine the `Payer` value that is set on a bucket by requesting the resource `requestPayment`.

To return the requestPayment resource

- Use a GET request to obtain the `requestPayment` resource, as shown in the following request.

```
GET ?requestPayment HTTP/1.1
Host: [BucketName].s3.amazonaws.com
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

If the request succeeds, Amazon S3 returns a response similar to the following.

```
HTTP/1.1 200 OK
x-amz-id-2: [id]
x-amz-request-id: [request_id]
Date: Wed, 01 Mar 2009 12:00:00 GMT
Content-Type: [type]
Content-Length: [length]
Connection: close
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<RequestPaymentConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
<Payer>Requester</Payer>
</RequestPaymentConfiguration>
```

This response shows that the `payer` value is set to `Requester`.

Downloading objects from Requester Pays buckets

Because requesters are charged for downloading data from Requester Pays buckets, the requests must contain a special parameter, `x-amz-request-payer`, which confirms that the requester knows that they will be charged for the download. To access objects in Requester Pays buckets, requests must include one of the following.

- For DELETE, GET, HEAD, POST, and PUT requests, include `x-amz-request-payer : requester` in the header
- For signed URLs, include `x-amz-request-payer=requester` in the request

If the request succeeds and the requester is charged, the response includes the header `x-amz-request-charged:requester`. If `x-amz-request-payer` is not in the request, Amazon S3 returns a 403 error and charges the bucket owner for the request.

> **Note**
>
> Bucket owners do not need to add `x-amz-request-payer` to their requests.
> Ensure that you have included `x-amz-request-payer` and its value in your signature calculation. For more information, see Using an Authorization Header in the *Amazon S3 API Reference* .

Using the REST API

To download objects from a Requester Pays bucket

- Use a GET request to download an object from a Requester Pays bucket, as shown in the following request.

```
GET / [destinationObject] HTTP/1.1
Host: [BucketName].s3.amazonaws.com
x-amz-request-payer : requester
Date: Wed, 01 Mar 2009 12:00:00 GMT
Authorization: AWS [Signature]
```

If the GET request succeeds and the requester is charged, the response includes `x-amz-request-charged:requester`.

Amazon S3 can return an `Access Denied` error for requests that try to get objects from a Requester Pays bucket. For more information, see Error Responses in the *Amazon Simple Storage Service API Reference*.

Using the AWS CLI

To download objects from a Requester Pays bucket using the AWS CLI, you specify `--request-payer requester` as part of your `get-object` request. For more information, see get-object in the *AWS CLI Reference*.

Working with objects in Amazon S3

To store your data in Amazon S3, you work with resources known as buckets and objects. A *bucket* is a container for objects. An *object* is a file and any metadata that describes that file.

To store an object in Amazon S3, you create a bucket and then upload the object to a bucket. When the object is in the bucket, you can open it, download it, and copy it. When you no longer need an object or a bucket, you can clean up these resources.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

> **Important**
>
> In the Amazon S3 console, when you choose **Open** or **Download As** for an object, these operations create presigned URLs. For the duration of five minutes, your object will be accessible to anyone who has access to these presigned URLs. For more information about presigned URLs, see Using presigned URLS.

With Amazon S3, you pay only for what you use. For more information about Amazon S3 features and pricing, see Amazon S3. If you are a new Amazon S3 customer, you can get started with Amazon S3 for free. For more information, see AWS Free Tier.

Topics

- Amazon S3 objects overview
- Naming Amazon S3 objects
- Working with object metadata
- Uploading objects
- Add preconditions to S3 operations with conditional requests
- Copying, moving, and renaming objects
- Downloading objects

- Checking object integrity in Amazon S3
- Deleting Amazon S3 objects
- Organizing, listing, and working with your objects
- Download and upload objects with presigned URLs
- Transforming objects with S3 Object Lambda
- Performing object operations in bulk with Batch Operations
- Querying data in place with Amazon S3 Select

Amazon S3 objects overview

Amazon S3 is an object store that uses unique key-values to store as many objects as you want. You store these objects in one or more buckets, and each object can be up to 50 TB in size. An object consists of the following:

Key

The name that you assign to an object. You use the object key to retrieve the object. For more information, see Working with object metadata.

Version ID

Within a bucket, a key and version ID uniquely identify an object. The version ID is a string that Amazon S3 generates when you add an object to a bucket. For more information, see Retaining multiple versions of objects with S3 Versioning.

Value

The content that you are storing.

An object value can be any sequence of bytes. Objects can range in size from zero to 50 TB. For more information, see Uploading objects.

> **Note**
>
> While Amazon S3 documentation commonly references 50 TB object size limit, the actual maximum object size is 53.7 TB (48.8 TiB). This limit is determined by multipart upload constraints: 10,000 maximum parts x 5 GiB per part = 50,000 GiB (53.7 TB)

Metadata

A set of name-value pairs with which you can store information regarding the object. You can assign metadata, referred to as user-defined metadata, to your objects in Amazon S3. Amazon S3 also assigns system-metadata to these objects, which it uses for managing objects. For more information, see Working with object metadata.

Subresources

Amazon S3 uses the subresource mechanism to store object-specific additional information. Because subresources are subordinates to objects, they are always associated with some other entity such as an object or a bucket. For more information, see Object subresources.

Access control information

You can control access to the objects you store in Amazon S3. Amazon S3 supports both the resource-based access control, such as an access control list (ACL) and bucket policies, and user-based access control. For more information about access control, see the following:

- Access control in Amazon S3
- Identity and Access Management for Amazon S3
- Configuring ACLs

Your Amazon S3 resources (for example, buckets and objects) are private by default. You must explicitly grant permission for others to access these resources. For more information about sharing objects, see Sharing objects with presigned URLs.

Tags

You can use tags to categorize your stored objects, for access control, or cost allocation. For more information, see Tagging your objects.

Object subresources

Amazon S3 defines a set of subresources associated with buckets and objects. Subresources are subordinates to objects. This means that subresources don't exist on their own. They are always associated with some other entity, such as an object or a bucket.

The following table lists the subresources associated with Amazon S3 objects.

Subresource	Description
acl	Contains a list of grants identifying the grantees and the permissions granted. When you create an object, the `acl` identifies the object owner as having full control over the object. You can retrieve an object ACL or replace it with an updated list of grants. Any update to an ACL requires you to replace the existing ACL. For more information about ACLs, see Access control list (ACL) overview.

Naming Amazon S3 objects

The *object key* (or key name) uniquely identifies the object in an Amazon S3 bucket. When you create an object, you specify the key name. For example, on the Amazon S3 console, when you select a bucket, a list of objects in your bucket appears. These names are the *object keys*.

The object key name consists of a sequence of Unicode characters encoded in UTF-8, with a maximum length of 1,024 bytes or approximately 1,024 Latin characters. In some locales, a single character may require 2 bytes for encoding. When naming your objects, be aware of the following:

- Object key names are case sensitive.
- Object key names include any prefixes (known as *folders* in the console). For example, `Development/Projects.xls` is the full object key name of the `Projects.xls` object located within the `Development` prefix (or folder). The prefix, the delimiter (/), and the name of the object are included in the 1,024 byte limitation for the object key name. For more information about prefixes and folders, see the section called “Choosing object key names”.
- Certain characters might require special handling when they're used in object key names. For more information, see the section called “Object key naming guidelines”.

> **Note**
>
> Object key names with the value `"soap"` aren't supported for virtual-hosted-style requests. For object key name values where `"soap"` is used, a path-style URL must be used instead.

Choosing object key names

The Amazon S3 data model is a flat structure: You create a bucket, and the bucket stores objects. There is no hierarchy of subbuckets or subfolders. However, you can infer logical hierarchy using key name prefixes and delimiters as the Amazon S3 console does. The Amazon S3 console supports a concept of folders. For more information about how to edit metadata from the Amazon S3 console, see Editing object metadata in the Amazon S3 console.

Suppose that your bucket (`admin-created`) has four objects with the following object keys:

`Development/Projects.xls`

`Finance/statement1.pdf`

`Private/taxdocument.pdf`

`s3-dg.pdf`

The console uses the key name prefixes (`Development/`, `Finance/`, and `Private/`) and delimiter (`/`) to present a folder structure. The `s3-dg.pdf` key doesn't contain a slash-delimited prefix, so its object appears directly at the root level of the bucket. If you open the `Development/` folder, you see the `Projects.xlsx` object in it.

- Amazon S3 supports buckets and objects, and there is no hierarchy. However, by using prefixes and delimiters in an object key name, the Amazon S3 console and the AWS SDKs can infer hierarchy and introduce the concept of folders.
- The Amazon S3 console implements folder object creation by creating a zero-byte object with the folder *prefix and delimiter* value as the key. These folder objects don't appear in the console. Otherwise they behave like any other objects and can be viewed and manipulated through the REST API, AWS CLI, and AWS SDKs.

Object key naming guidelines

You can use any UTF-8 character in an object key name. However, using certain characters in key names can cause problems with some applications and protocols. The following guidelines help you maximize compliance with DNS, web-safe characters, XML parsers, and other APIs.

Safe characters

The following character sets are generally safe for use in key names:

Alphanumeric characters	• 0-9 • a-z • A-Z
Special characters	• Exclamation point (!) • Hyphen (-) • Underscore (_) • Period (.) • Asterisk (*) • Single quotation mark (') • Opening parenthesis (() • Closing parenthesis ())

The following are examples of valid object key names:

- `4my-organization`
- `my.great_photos-2014/jan/myvacation.jpg`
- `videos/2014/birthday/video1.wmv`

Note

If you use the Amazon S3 console to download objects that have key names that end with periods (`.`), the periods are removed from the ends of the key names of the downloaded objects. To retain periods at the ends of key names in downloaded objects, you must use the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.
In addition, be aware of the following prefix limitations:

- Objects with a prefix of `./` must be uploaded or downloaded with the AWS CLI, AWS SDKs, or REST API. You can't use the Amazon S3 console to upload these objects.
- Object keys that contain relative path elements (for example, `../`) are valid if, when parsed left-to-right, the cumulative count of relative path segments never exceeds the number of non-relative path elements encountered. This rule applies to all requests made by using the Amazon S3 console, Amazon S3 REST API, AWS CLI, and AWS SDKs.

For example:

- `videos/2014/../../video1.wmv` is valid.
- `videos/../../video1.wmv` isn't valid.
- `videos/../../2014/video1.wmv` isn't valid.

Period-only path segments

Object keys containing period-only path segments (`.` or `..`) can cause unexpected behavior when processed by applications, SDKs, or tools that interpret these as relative path references.

The following patterns can cause issues:

- `folder/./file.txt` - Contains current directory reference
- `folder/../file.txt` - Contains parent directory reference
- `./file.txt` - Starts with current directory reference
- `../file.txt` - Starts with parent directory reference

The following patterns work normally:

- `folder/.hidden/file.txt` - Period is part of filename, not standalone
- `folder/..backup/file.txt` - Periods are part of filename, not standalone

When applications process object keys with period-only segments, the following behavioral impacts can occur:

- *Path normalization* - Many systems automatically resolve `.` and `..` references, potentially changing the effective path (for example, `folder/./file.txt` becomes `folder/file.txt`)
- *Access issues* - Applications might fail to locate objects due to path resolution differences
- *Inconsistent behavior* - Different tools and SDKs might handle these patterns differently

Important

To avoid these issues, we recommend avoiding period-only path segments in object key names. Use alternative naming conventions for organizational purposes.

Characters that might require special handling

The following characters in a key name might require additional code handling and most likely must be URL encoded or referenced as HEX. Some of these characters are non-printable characters that your browser might not handle, which also require special handling:

- Ampersand (&)
- Dollar sign ($)
- ASCII character ranges 00–1F hex (0–31 decimal) and 7F (127 decimal)
- At symbol (@)
- Equal sign (=)
- Semicolon (;)
- Forward slash (/)
- Colon (:)
- Plus sign (+)
- Space – Significant sequences of spaces might be lost in some cases (especially multiple spaces)
- Comma (,)
- Question mark (?)

Characters to avoid

We recommend not using the following characters in a key name because of significant special character handling, which isn't consistent across all applications:

- Backslash (\)
- Left brace ({)
- Non-printable ASCII characters (128–255 decimal characters)
- Caret or circumflex (^)

- Right brace (})
- Percent character (%)
- Grave accent or backtick (`)
- Right bracket (])
- Quotation mark (")
- Greater than sign (>)
- Left bracket ([)
- Tilde (~)
- Less than sign (<)
- Pound sign (#)
- Vertical bar or pipe (|)

XML-related object key constraints

As specified by the XML standard on end-of-line handling, all XML text is normalized such that single carriage returns (ASCII code 13) and carriage returns immediately followed by a line feed (ASCII code 10), also known as newline characters, are replaced by a single line feed character. To ensure the correct parsing of object keys in XML requests, carriage returns and other special characters must be replaced with their equivalent XML entity code when they're inserted within XML tags.

The following is a list of such special characters and their equivalent XML entity codes:

- Apostrophe (') must be replaced with `'`
- Quotation mark (") must be replaced with `"`
- Ampersand (&) must be replaced with `&`
- Less than sign (<) must be replaced with `<`
- Greater than sign (>) must be replaced with `>`
- Carriage return (\r) must be replaced with `
` or `
`
- Newline (\n) must be replaced with `
` or `
`

Example

The following example illustrates the use of an XML entity code as a substitution for a carriage return. This `DeleteObjects` request deletes an object with the `key` parameter `/some/prefix/objectwith\rcarriagereturn` (where the `\r` is the carriage return).

```
<Delete xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Object>
    <Key>/some/prefix/objectwith&#13;carriagereturn</Key>
  </Object>
</Delete>
```

Object key sort order

Amazon S3 sorts object keys, including prefixes, lexicographically by their UTF-8 encoded byte values.

ASCII characters sort in the following order:

- Special characters (for example, `!`, `/`)
- Uppercase letters (A–Z)
- Lowercase letters (a–z)

Non-ASCII characters (for example, é, 中 文) are encoded as multi-byte UTF-8 sequences and typically sort after ASCII characters because of their higher byte values (for example, `0xC3` for é, `0xE4` for 中).

For example, prefixes such as `apple/`, `Apple/`, `éclair/`, `# #/` would sort as:

1. `Apple/` (starts with `0x41`)

2. `apple/` (starts with `0x61`)

3. `éclair/` (starts with `0xC3 0xA9`)

4. `# #/` (starts with `0xE4 0xB8 0xAD 0xE6 0x96 0x87`)

Working with object metadata

Amazon S3 provides several ways to attach metadata to your objects. *Annotations* let you attach rich data payloads of up to 1 MB to objects using dedicated API operations. *Object tags* are key-value pairs for categorizing and controlling access to objects. You can also set *user-defined metadata* at upload time, and Amazon S3 maintains *system-defined metadata* automatically. For more information, see the section called “Custom object metadata”, the section called “Annotating objects”, the section called “Tagging objects”, the section called “User-defined metadata”, and the section called “System-defined object metadata”.

When you create an object, you specify the *object key* (or *key name*), which uniquely identifies the object in an Amazon S3 bucket. For more information, see Naming Amazon S3 objects. You can also set user-defined metadata in Amazon S3 at the time that you upload the object.

After you upload the object, you can't modify this user-defined metadata. The only way to modify this metadata is to make a copy of the object and set the metadata. For more information about editing metadata by using the Amazon S3 console, see Editing object metadata in the Amazon S3 console.

Custom object metadata

Custom object metadata is metadata that you control. You can use custom object metadata to store information about your objects that is meaningful to your applications. Amazon S3 supports three types of custom object metadata: annotations, object tags, and user-defined metadata.

Annotations

Annotations are named data payloads that you can attach to any Amazon S3 object after it's uploaded. Each annotation can store up to 1 MB of UTF-8 text, such as AI-generated output, classification results, or processing records. You can create, retrieve, list, and delete annotations without modifying the object itself. For more information, see the section called “Annotating objects”.

Object tags

Object tags are key-value pairs that you can use to categorize, organize, and control access to your objects. Each object can have up to 10 tags. You can use tags in IAM and bucket policies, S3 Lifecycle rules, and cost allocation reports. For more information, see the section called “Tagging objects”.

User-defined metadata

When uploading an object, you can also assign metadata to the object. You provide this optional information as a name-value (key-value) pair when you send a PUT or POST request to create the object. When you upload objects using the REST API, the optional user-defined metadata names must begin with `x-amz-meta-` to distinguish them from other HTTP headers. When you retrieve the object using the REST API, this prefix is returned. When you upload objects using the SOAP API, the prefix is not required. When you retrieve the object using the SOAP API, the prefix is removed, regardless of which API you used to upload the object.

> **Note**
>
> SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the AWS SDKs.

When metadata is retrieved through the REST API, Amazon S3 combines headers that have the same name (ignoring case) into a comma-delimited list. If some metadata contains unprintable characters, it is not returned. Instead, the `x-amz-missing-meta` header is returned with a value of the number of unprintable metadata entries. The `HeadObject` action retrieves metadata from an object without returning the object itself. This operation is useful if you're only interested in an object's metadata. To use HEAD, you must have READ access to the object. For more information, see HeadObject in the *Amazon Simple Storage Service API Reference*.

User-defined metadata is a set of key-value pairs. Amazon S3 stores user-defined metadata keys in lowercase.

Amazon S3 allows arbitrary Unicode characters in your metadata values.

To avoid issues related to the presentation of these metadata values, you should conform to using US-ASCII characters when using REST and UTF-8 when using SOAP or browser-based uploads through POST.

When using non-US-ASCII characters in your metadata values, the provided Unicode string is examined for non-US-ASCII characters. Values of such headers are character decoded as per RFC 2047 before storing and encoded as per RFC 2047 to make them mail-safe before returning. If the string contains only US-ASCII characters, it is presented as is.

The following is an example.

```
PUT /Key HTTP/1.1
Host: amzn-s3-demo-bucket.s3.amazonaws.com
x-amz-meta-nonascii: ÄMÄZÕÑ S3

HEAD /Key HTTP/1.1
Host: amzn-s3-demo-bucket.s3.amazonaws.com
x-amz-meta-nonascii: =?UTF-8?B?w4PChE3Dg8KEWsODwpXDg8KRIFMz?=

PUT /Key HTTP/1.1
Host: amzn-s3-demo-bucket.s3.amazonaws.com
x-amz-meta-ascii: AMAZONS3

HEAD /Key HTTP/1.1
Host: amzn-s3-demo-bucket.s3.amazonaws.com
x-amz-meta-ascii: AMAZONS3
```

Note

The PUT request header is limited to 8 KB in size. Within the PUT request header, the user-defined metadata is limited to 2 KB in size. The size of user-defined metadata is measured by taking the sum of the number of bytes in the UTF-8 encoding of each key and value.

For information about changing the metadata of your object after it has been uploaded by creating a copy of the object, modifying it, and replacing the old object, or creating a new version, see Editing object metadata in the Amazon S3 console.

Query your metadata and accelerate data discovery with S3 Metadata

To easily find, store, and query metadata for your S3 objects, you can use S3 Metadata. With S3 Metadata, you can quickly prepare data for use in business analytics, content retrieval, artificial intelligence and machine learning (AI/ML) model training, and more.

S3 Metadata accelerates data discovery by automatically capturing metadata for the objects in your general purpose buckets and storing it in read-only, fully managed Apache Iceberg tables that you can query. These read-only tables are called *metadata tables*. As objects are added to,

updated, and removed from your general purpose buckets, S3 Metadata automatically refreshes the corresponding metadata tables to reflect the latest changes.

By default, S3 Metadata provides system-defined object metadata, such as an object's creation time and storage class, and custom metadata, such as tags, user-defined metadata, and annotations. S3 Metadata also provides event metadata, such as when an object is updated or deleted, and the AWS account that made the request.

Metadata tables are stored in S3 table buckets, which provide storage that's optimized for tabular data. To query your metadata, you can integrate your table bucket with AWS analytics services, such as Amazon Athena, Amazon Redshift, and AWS Quick.

For more information about S3 Metadata, see the section called "Discovering your data with S3 Metadata tables".

System-defined object metadata

For each object stored in a bucket, Amazon S3 maintains a set of system metadata. Amazon S3 processes this system metadata as needed. For example, Amazon S3 maintains object-creation date and size metadata, using this information as part of object management.

There are two categories of system metadata:

- **System controlled** – Metadata such as the object-creation date is system controlled, which means that only Amazon S3 can modify the date value.
- **User controlled** – Other system metadata, such as the storage class configured for the object and whether the object has server-side encryption enabled, are examples of system metadata whose values you control. If your bucket is configured as a website, sometimes you might want to redirect a page request either to another page or an external URL. In this case, a webpage is an object in your bucket. Amazon S3 stores the page redirect value as system metadata, which you can control.

  When you create objects, you can configure the values of these system metadata items or update the values when you need to. For more information about storage classes, see Understanding and managing Amazon S3 storage classes.

  Amazon S3 uses AWS KMS keys to encrypt your Amazon S3 objects. AWS KMS encrypts only the object data. The checksum and the specified algorithm are stored as part of the object's metadata. If server-side encryption is requested for the object, then the checksum is stored in

encrypted form. For more information about server-side encryption, see Protecting data with encryption.

> **Note**
>
> The PUT request header is limited to 8 KB in size. Within the PUT request header, the system-defined metadata is limited to 2 KB in size. The size of system-defined metadata is measured by taking the sum of the number of bytes in the US-ASCII encoding of each key and value.

The following table provides a list of system-defined metadata and whether you can update it.

Name	Description	Can user modify the value?
`Date`	The current date and time.	No
`Cache-Control`	A general header field used to specify caching policies.	Yes
`Content-Disposition`	Object presentational information.	Yes
`Content-Encoding`	The content encodings (like compression) that have been applied to the object's data	Yes
`Content-Length`	The object size in bytes.	No
`Content-Type`	The object type.	Yes
`Last-Modified`	The object creation date or the last modified date, whichever is the latest. For multipart uploads, the object creation date is the date of initiation of the multipart upload.	No
`ETag`	An entity tag (ETag) that represents a specific version of an object. For objects that are not uploaded as	No

Name	Description	Can user modify the value?
	a multipart upload and are either unencrypted or encrypted by server-side encryption with Amazon S3 managed keys (SSE-S3), the ETag is an MD5 digest of the data.	
`x-amz-server-side-encryption`	A header that indicates whether server-side encryption is enabled for the object, and whether that encryption is using the AWS Key Management Service (AWS KMS) keys (SSE-KMS) or using Amazon S3 managed encryption keys (SSE-S3). For more information, see Protecting data with server-side encryption.	Yes
`x-amz-checksum-crc64nvme` , `x-amz-checksum-crc32` , `x-amz-checksum-crc32c` , `x-amz-checksum-sha1` , `x-amz-checksum-sha256`	Headers that contain the checksum or digest of the object. At most, one of these headers will be set at a time, depending on the checksum algorithm that you instruct Amazon S3 to use. For more information about choosing the checksum algorithm, see Checking object integrity in Amazon S3.	No
`x-amz-checksum-type`	The checksum type, which determines how part-level checksums are combined to create an object-level checksum for multipart objects.	Yes
`x-amz-version-id`	The object version. When you enable versioning on a bucket, Amazon S3 assigns a version ID to objects added to the bucket. For more information, see Retaining multiple versions of objects with S3 Versioning.	No

Name	Description	Can user modify the value?
`x-amz-delete-marker`	A Boolean marker that indicates whether the object is a delete marker. This marker is used only in buckets that have versioning enabled,	No
`x-amz-storage-class`	The storage class used for storing the object. For more information, see Understanding and managing Amazon S3 storage classes.	Yes
`x-amz-website-redirect-location`	A header that redirects requests for the associated object to another object in the same bucket or to an external URL. For more information, see (Optional) Configuring a webpage redirect.	Yes
`x-amz-server-side-encryption-aws-kms-key-id`	A header that indicates the ID of the AWS KMS symmetric encryption KMS key that was used to encrypt the object. This header is used only when the `x-amz-server-side-encryption` header is present and has the value of `aws:kms`.	Yes
`x-amz-server-side-encryption-customer-algorithm`	A header that indicates whether server-side encryption with customer-provided encryption keys (SSE-C) is enabled. For more information, see Using server-side encryption with customer-provided keys (SSE-C).	Yes
`x-amz-tagging`	The tag-set for the object. The tag-set must be encoded as URL Query parameters.	Yes

Annotating your objects

Use annotations to attach named data payloads to your Amazon S3 objects. Each annotation is a custom metadata payload between 1 byte and 1 MiB in size that you can create, retrieve, list, and delete without modifying the object itself.

You can associate up to 1,000 annotations with an object version. Each annotation has a unique name and can store structured data such as AI-generated labels, document context, processing results, or compliance records.

Common use cases include storing machine learning inference results, AI-generated embeddings, content moderation labels, document classification output, data lineage and audit trails, compliance labels such as PII flags or retention policies, medical imaging metadata, digital asset rights information, and ETL pipeline status alongside the source object.

You manage annotations using dedicated API operations, so you don't need to re-upload the object to add or update metadata.

You can enable an annotation table as part of your S3 Metadata configuration to query annotation data at scale using Athena and other analytics services. S3 Metadata stores annotation data in fully managed Apache Iceberg tables that Amazon S3 automatically keeps up to date. For more information, see the section called “Discovering your data with S3 Metadata tables”.

Annotations are available in all commercial AWS Regions, the AWS GovCloud (US) Regions, and the China Regions (Beijing and Ningxia). Annotations are not available in the Middle East (UAE) and Middle East (Bahrain) Regions. S3 Metadata annotation tables are available in all Regions where S3 Metadata is available.

When to use annotations vs. object tags

Use the following comparison to determine whether annotations or object tags are the best fit for your use case.

Characteristic	Object tags	Annotations
Maximum per object	10 per object version	1,000 per object version
Maximum size	128 chars (key) + 256 chars (value)	512 bytes (name) + 1 MiB (payload)
Data format	Key-value string pairs	Any UTF-8 text (JSON, XML, YAML, etc.)
Mutability	Yes (PutObjectTagging)	Yes (PutObjectAnnotation)

Characteristic	Object tags	Annotations
Set during upload	Yes (PutObject, POST)	No (PutObjectAnnotation only, after upload)

Choose annotations when you need to store structured data (such as JSON or XML), payloads larger than 256 characters, or more than 10 metadata entries per object. Choose object tags when you need IAM policy integration, Amazon S3 Lifecycle rule filtering, or cost allocation reporting.

API operations for annotations

Amazon S3 supports the following API operations for working with annotations:

- **PutObjectAnnotation** – Creates or overwrites an annotation on an object. You specify the annotation name and payload in the request.
- **GetObjectAnnotation** – Returns the payload of a specific annotation by name.
- **ListObjectAnnotations** – Returns the list of annotations on an object. The response includes each annotation's name, size, ETag, and last modified date.
- **DeleteObjectAnnotation** – Removes a specific annotation by name.

Amazon S3 also supports annotations in the following API operations:

- **CopyObject** – Copies annotations from the source object by default. You can specify the `x-amz-annotation-directive` header to control whether annotations are copied (`COPY`) or excluded (`EXCLUDE`).
- **UpdateBucketMetadataAnnotationTableConfiguration** – Enables or disables the annotation table in your S3 Metadata configuration.
- **CreateBucketMetadataConfiguration** – Accepts a new `AnnotationTableConfiguration` parameter to enable annotation tables when you create an S3 Metadata configuration.
- **GetBucketMetadataConfiguration** – Returns `AnnotationTableConfigurationResult` in the response, which indicates the current status of the annotation table.

Annotation limits

Each object version supports up to 1,000 annotations. Annotations that are associated with an object version must have unique annotation names. The following limits apply:

- An annotation name can be up to 512 bytes in length (UTF-8), subject to the naming rules below.
- An annotation payload must be between 1 byte and 1 MiB in size.
- The total annotation storage per object can be up to 1 GiB (1,000 annotations at 1 MiB each).
- Supported checksum algorithms: CRC32, CRC32C, CRC64NVME, SHA1, SHA256, SHA512, XXHASH64, XXHASH3, XXHASH128.

Annotation naming rules

Annotation names must meet the following requirements:

- Must be between 1 and 512 bytes in length.
- Can contain only the following characters: letters (any language), digits (0-9), underscore (_), period (.), and hyphen (-).
- Cannot start with `aws` or `s3` (case-insensitive). For example, `aws`, `AWS`, `s3`, and `S3` are all reserved prefixes.
- Must not be empty or consist only of whitespace.

Encryption

Annotations are automatically encrypted at rest using the same encryption configuration as the parent object. The encryption type is inherited from the parent object, not the bucket default.

- **SSE-S3** – If the parent object uses server-side encryption with Amazon S3 managed keys (SSE-S3), annotations are encrypted with SSE-S3. If the parent object has no server-side encryption configured, annotations are encrypted with SSE-S3 by default.
- **SSE-KMS** – If the parent object uses server-side encryption with AWS KMS keys (SSE-KMS), annotations are encrypted with the same KMS key. This applies to both customer managed keys and AWS managed keys. S3 Bucket Keys are supported.
- **DSSE-KMS** – If the parent object uses dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), annotations are encrypted with DSSE-KMS using the same key.
- **SSE-C** – Server-side encryption with customer-provided keys (SSE-C) is not supported for annotations. If you attempt to add an annotation to an object encrypted with SSE-C, Amazon S3 returns an error.

Checksums

When you upload an annotation using `PutObjectAnnotation`, you can provide a checksum to verify data integrity. The checksum algorithm for an annotation is independent from the parent object's checksum algorithm.

When you copy an object using `CopyObject`, Amazon S3 preserves the annotation checksum values from the source. If you specify a different checksum algorithm in the copy request, the new algorithm applies to both the object and its annotations.

Supported algorithms: CRC32, CRC32C, CRC64NVME, SHA1, SHA256, SHA512, XXHASH64, XXHASH3, XXHASH128.

If the annotation doesn't have a specified checksum algorithm or checksum value, Amazon S3 uses the CRC-64/NVME algorithm to calculate the checksum value for the annotation.

Versioning behavior

Annotations are attached to a specific object version.

Annotations of one object version are independent of annotations on other versions of the same object. Creating a new version does not copy annotations from the previous version. Deleting or adding an annotation on one version does not affect annotations on other versions. Overwriting an object replaces its annotations with whatever annotations the new version has (if none, effectively deletes them).

Adding, updating, or removing an annotation does not modify the parent object's ETag.

In a non-versioned bucket, if you delete or overwrite the object, the annotations are deleted with it.

In a versioned bucket, the following behavior applies:

- A simple DELETE request (without specifying a version ID) creates a delete marker but preserves annotations on the underlying version.
- Deleting a specific version ID deletes that version and all associated annotations.
- Annotations are not independently versioned. When you overwrite an annotation with the same name, Amazon S3 replaces the previous value without creating a new object version.

Important

Annotation deletion is permanent and irreversible, even in a versioned bucket. Unlike objects in versioned buckets, annotations do not have delete markers or version history. Once you delete an annotation, it cannot be recovered.

Copy behavior and consistency

When you copy an object using the `CopyObject` API (for objects smaller than 5 GiB), Amazon S3 copies annotations along with the object in a single operation.

When you copy objects using multipart upload (for example, when the AWS CLI or AWS SDKs use Transfer Manager for objects larger than approximately 8 MB), annotations are not copied by default. To include annotations, specify `--copy-props all` in the AWS CLI or the equivalent SDK configuration. With this opt-in, the SDK reads source annotations, completes the multipart upload, and then writes each annotation to the destination. Between the upload completion and the last annotation write, the destination object exists without all its annotations.

Considerations

- You cannot add annotations as part of a `PutObject` or multipart upload request. To add annotations to an object, call `PutObjectAnnotation` after the object is uploaded. To copy an existing object with its annotations to a new location, use `CopyObject` with the default annotation directive.
- To add or update annotations across many objects in bulk, use Batch Operations to invoke an Lambda function that calls `PutObjectAnnotation` on each object. For more information, see the section called “Invoke AWS Lambda function”.
- Annotations are not supported by the following features: S3 Inventory Reports, API Gateway, S3 Storage Lens, Amazon S3 File Gateway, Amazon FSx, S3 on Outposts, S3 Express One Zone (directory buckets), and Amazon S3 Files.
- To ensure you are writing annotations to the current version of an object and not one that has been overwritten, use the `x-amz-object-if-match` conditional header with `PutObjectAnnotation` or `DeleteObjectAnnotation`. This header validates the parent object's ETag to confirm the object has not been overwritten since the caller last read it. Adding tags or annotations does not change the ETag.

- You cannot add an annotation conditionally on the presence or absence of another annotation. The `x-amz-object-if-match` header validates the parent object's ETag only, not annotation state.
- Annotation payloads must be valid UTF-8 encoded text. To store binary data, encode the data with Base64 before writing the annotation.
- You can call annotation API operations (`PutObjectAnnotation`, `GetObjectAnnotation`, `ListObjectAnnotations`, `DeleteObjectAnnotation`) on objects in any storage class, including S3 Glacier and S3 Glacier Deep Archive, without restoring the object first.

Additional configurations

This section explains how annotations relate to other configurations.

Replication

If you have S3 Replication configured on your bucket, Amazon S3 replicates annotations automatically. Each annotation replicates independently. For more information, see the section called “What's replicated?”.

To replicate annotations, add `s3:GetObjectVersionAnnotationForReplication` to the source bucket permissions in your replication IAM role. For more information, see the section called “Setting up permissions”.

To prevent annotation replication while allowing object replication, add a deny statement for `s3:ReplicateObjectAnnotation` in the replication role policy. Object replication continues to succeed; only annotation replication is blocked.

Event notifications

Amazon S3 can send event notifications when annotations are created, updated, or deleted. You can configure the following event types:

- `s3:ObjectAnnotation:Put` – Sent when an annotation is created or updated.
- `s3:ObjectAnnotation:Delete` – Sent when an annotation is deleted.

For more information, see the section called “Notification types and destinations”.

Managing annotations

You can add, retrieve, and delete annotations on your Amazon S3 objects using the AWS Management Console, AWS CLI, or AWS SDKs. The following sections describe how to perform each operation.

Adding or updating an annotation

You can add a new annotation to an object or update an existing annotation by replacing its payload.

Using the S3 console

To add an annotation to an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object that you want to annotate.
4. Choose the **Properties** tab.
5. Scroll to the **Annotations** section and choose **Add annotation**.
6. For **Name**, enter a name for the annotation (for example, `ml.sentiment`). Names must be 1–512 bytes and can contain Unicode letters, digits, underscores, periods, and hyphens. Names cannot start with `aws` or `s3`.
7. Choose **Choose file** to upload the annotation content. The file must be UTF-8 encodable and up to 1 MB in size.

To complete the annotation

1. (Optional) Under **Checksums**, choose a checksum algorithm.
2. Choose **Add annotation**.

> **Note**
>
> To view the content of an annotation after it is created, you must download the annotation. The S3 console does not display annotation content inline.

Using the AWS CLI

To add or update an annotation on an object, use the `put-object-annotation` command. The `--annotation-payload` parameter accepts a file path.

```
aws s3api put-object-annotation \
    --bucket amzn-s3-demo-bucket \
    --key myDocument.pdf \
    --annotation-name "ml.sentiment" \
    --annotation-payload /tmp/sentiment.json
```

To write the annotation only if the object has not been overwritten since you last read it, specify the `--object-if-match` parameter with the object's current ETag.

```
aws s3api put-object-annotation \
    --bucket amzn-s3-demo-bucket \
    --key myDocument.pdf \
    --annotation-name "ml.sentiment" \
    --annotation-payload /tmp/sentiment.json \
    --object-if-match "\"d6eb32081c822ed572b70567826d9d9d\""
```

Using the AWS SDKs

The following example uses the AWS SDK for Python (Boto3) to add an annotation to an object.

```
import boto3
import json

s3 = boto3.client('s3')

s3.put_object_annotation(
    Bucket='amzn-s3-demo-bucket',
    Key='myDocument.pdf',
    AnnotationName='ml.sentiment',
    AnnotationPayload=json.dumps({
        'score': 0.95,
        'label': 'positive',
        'model': 'sentiment-v3'
    }).encode('utf-8')
)
```

Retrieving annotations

You can retrieve a specific annotation or list all annotations on an object.

Using the S3 console

To retrieve an annotation

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object.
4. Choose the **Properties** tab and scroll to the **Annotations** section.
5. To view annotation content, choose **Download** next to the annotation name.

Using the AWS CLI

To retrieve a specific annotation, use the `get-object-annotation` command.

```
aws s3api get-object-annotation \
    --bucket amzn-s3-demo-bucket \
    --key myDocument.pdf \
    --annotation-name "ml.sentiment" \
    output.json
```

To list all annotations on an object, use the `list-object-annotations` command.

```
aws s3api list-object-annotations \
    --bucket amzn-s3-demo-bucket \
    --key myDocument.pdf
```

Using the AWS SDKs

The following example uses the AWS SDK for Python (Boto3) to retrieve and list annotations.

```
import boto3
import json

s3 = boto3.client('s3')
```

```
# Retrieve a specific annotation
response = s3.get_object_annotation(
    Bucket='amzn-s3-demo-bucket',
    Key='myDocument.pdf',
    AnnotationName='ml.sentiment'
)
content = json.loads(response['AnnotationPayload'].read())

# List all annotations on an object
response = s3.list_object_annotations(
    Bucket='amzn-s3-demo-bucket',
    Key='myDocument.pdf'
)
for annotation in response['Annotations']:
    print(f"{annotation['AnnotationName']} - {annotation['Size']} bytes")
```

Deleting annotations

You can delete a specific annotation from an object. Annotation deletion is permanent and cannot be undone.

Using the S3 console

To delete an annotation

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object.
4. Choose the **Properties** tab and scroll to the **Annotations** section.
5. Select the annotation that you want to delete.
6. Choose **Delete**, then confirm the deletion.

Using the AWS CLI

To delete a specific annotation, use the `delete-object-annotation` command.

```
aws s3api delete-object-annotation \
    --bucket amzn-s3-demo-bucket \
    --key myDocument.pdf \
```

```
    --annotation-name "ml.sentiment"
```

Using the AWS SDKs

The following example uses the AWS SDK for Python (Boto3) to delete an annotation.

```
import boto3

s3 = boto3.client('s3')

s3.delete_object_annotation(
    Bucket='amzn-s3-demo-bucket',
    Key='myDocument.pdf',
    AnnotationName='ml.sentiment'
)
```

Tagging your objects

Use object tagging to categorize storage. Each tag is a key-value pair.

If you need to store structured data larger than 256 characters per value, associate more than 10 metadata entries with an object, or attach rich content such as machine learning inference results or AI-generated labels, see Annotating your objects.

You can add tags to new objects when you upload them, or you can add them to existing objects.

- You can associate up to 10 tags with an object. Tags that are associated with an object must have unique tag keys.
- A tag key can be up to 128 Unicode characters in length, and tag values can be up to 256 Unicode characters in length. Amazon S3 object tags are internally represented in UTF-16. Note that in UTF-16, characters consume either 1 or 2 character positions.
- The key and values are case sensitive.
- For more information about tag restrictions, see User-defined tag restrictions in the *AWS Billing and Cost Management User Guide*. For basic tag restrictions, see Tag restrictions in the *Amazon EC2 User Guide*.

Examples

Consider the following tagging examples:

Example PHI information

Suppose that an object contains protected health information (PHI) data. You might tag the object using the following key-value pair.

```
PHI=True
```

or

```
Classification=PHI
```

Example Project files

Suppose that you store project files in your S3 bucket. You might tag these objects with a key named `Project` and a value, as shown following.

```
Project=Blue
```

Example Multiple tags

You can add multiple tags to an object, as shown following.

```
Project=x
Classification=confidential
```

Key name prefixes and tags

Object key name prefixes also enable you to categorize storage. However, prefix-based categorization is one-dimensional. Consider the following object key names:

```
photos/photo1.jpg
project/projectx/document.pdf
project/projecty/document2.pdf
```

These key names have the prefixes `photos/`, `project/projectx/`, and `project/projecty/`. These prefixes enable one-dimensional categorization. That is, everything under a prefix is one category. For example, the prefix `project/projectx` identifies all documents related to project x.

With tagging, you now have another dimension. If you want photo1 in project x category, you can tag the object accordingly.

Additional benefits

In addition to data classification, tagging offers benefits such as the following:

- Object tags enable fine-grained access control of permissions. For example, you could grant a user permissions to read-only objects with specific tags.
- Object tags enable fine-grained object lifecycle management in which you can specify a tag-based filter, in addition to a key name prefix, in a lifecycle rule.
- When using Amazon S3 analytics, you can configure filters to group objects together for analysis by object tags, by key name prefix, or by both prefix and tags.
- You can also customize Amazon CloudWatch metrics to display information by specific tag filters. The following sections provide details.

> **Important**
>
> It is acceptable to use tags to label objects containing confidential data, such as personally identifiable information (PII) or protected health information (PHI). However, the tags themselves shouldn't contain any confidential information.

Adding object tag sets to multiple Amazon S3 object with a single request

To add object tag sets to more than one Amazon S3 object with a single request, you can use S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on. S3 Batch Operations calls the respective API operation to perform the specified operation. A single Batch Operations job can perform the specified operation on billions of objects containing exabytes of data.

The S3 Batch Operations feature tracks progress, sends notifications, and stores a detailed completion report of all actions, providing a fully managed, auditable, serverless experience. You can use S3 Batch Operations through the Amazon S3 console, AWS CLI, AWS SDKs, or REST API. For more information, see the section called “Batch Operations basics”.

For more information about object tags, see Managing object tags.

API operations related to object tagging

Amazon S3 supports the following API operations that are specifically for object tagging:

Object API operations

- PUT Object tagging – Replaces tags on an object. You specify tags in the request body. There are two distinct scenarios of object tag management using this API.
  - Object has no tags – Using this API you can add a set of tags to an object (the object has no prior tags).
  - Object has a set of existing tags – To modify the existing tag set, you must first retrieve the existing tag set, modify it on the client side, and then use this API to replace the tag set.

    > **Note**
    >
    > If you send this request with an empty tag set, Amazon S3 deletes the existing tag set on the object. If you use this method, you will be charged for a Tier 1 Request (PUT). For more information, see Amazon S3 Pricing.
    > The DELETE Object tagging request is preferred because it achieves the same result without incurring charges.

- GET Object tagging – Returns the tag set associated with an object. Amazon S3 returns object tags in the response body.
- DELETE Object tagging – Deletes the tag set associated with an object.

Other API operations that support tagging

- PUT Object and Initiate Multipart Upload– You can specify tags when you create objects. You specify tags using the `x-amz-tagging` request header.
- GET Object – Instead of returning the tag set, Amazon S3 returns the object tag count in the `x-amz-tag-count` header (only if the requester has permissions to read tags) because the header response size is limited to 8 K bytes. If you want to view the tags, you make another request for the GET Object tagging API operation.
- POST Object – You can specify tags in your POST request.

  As long as the tags in your request don't exceed the 8 K byte HTTP request header size limit, you can use the `PUT Object` API to create objects with tags. If the tags you specify exceed the header size limit, you can use this POST method in which you include the tags in the body.

PUT Object - Copy – You can specify the `x-amz-tagging-directive` in your request to direct Amazon S3 to either copy (default behavior) the tags or replace tags by a new set of tags provided in the request.

Note the following:

- S3 Object Tagging is strongly consistent. For more information, see Amazon S3 data consistency model.

Additional configurations

This section explains how object tagging relates to other configurations.

Object tagging and lifecycle management

In bucket lifecycle configuration, you can specify a filter to select a subset of objects to which the rule applies. You can specify a filter based on the key name prefixes, object tags, or both.

Suppose that you store photos (raw and the finished format) in your Amazon S3 bucket. You might tag these objects as shown following.

```
phototype=raw
or
phototype=finished
```

You might consider archiving the raw photos to Amazon Glacier sometime after they are created. You can configure a lifecycle rule with a filter that identifies the subset of objects with the key name prefix (`photos/`) that have a specific tag (`phototype=raw`).

For more information, see Managing the lifecycle of objects.

Object tagging and replication

If you configured Replication on your bucket, Amazon S3 replicates tags, provided you grant Amazon S3 permission to read the tags. For more information, see Setting up live replication overview.

Object tagging event notifications

You can set up an Amazon S3 event notification to receive notice when an object tag is added or deleted from an object. The `s3:ObjectTagging:Put` event type notifies you when a tag is PUT on an object or when an existing tag is updated. The `s3:ObjectTagging:Delete` event type notifies you when a tag is removed from an object. For more information, see Enabling event notifications.

For more information about object tagging, see the following topics:

Topics

- Tagging and access control policies
- Managing object tags

Tagging and access control policies

You can also use permissions policies (bucket and user policies) to manage permissions related to object tagging. For policy actions see the following topics:

- Object operations
- Bucket operations

Object tags enable fine-grained access control for managing permissions. You can grant conditional permissions based on object tags. Amazon S3 supports the following condition keys that you can use to grant conditional permissions based on object tags:

- `s3:ExistingObjectTag/<tag-key>` – Use this condition key to verify that an existing object tag has the specific tag key and value.

  > **Note**
  >
  > When granting permissions for the `PUT Object` and `DELETE Object` operations, this condition key is not supported. That is, you cannot create a policy to grant or deny a user permissions to delete or overwrite an object based on its existing tags.

- `s3:RequestObjectTagKeys` – Use this condition key to restrict the tag keys that you want to allow on objects. This is useful when adding tags to objects using the PutObjectTagging and PutObject, and POST object requests.

- `s3:RequestObjectTag/<tag-key>` – Use this condition key to restrict the tag keys and values that you want to allow on objects. This is useful when adding tags to objects using the PutObjectTagging and PutObject, and POST Bucket requests.

For a complete list of Amazon S3 service-specific condition keys, see Bucket policy examples using condition keys. The following permissions policies illustrate how object tagging enables fine grained access permissions management.

Example 1: Allow a user to read only the objects that have a specific tag and key value

The following permissions policy limits a user to only reading objects that have the `environment:production` tag key and value. This policy uses the `s3:ExistingObjectTag` condition key to specify the tag key and value.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
  {
    "Principal": {
      "AWS": [
        "arn:aws:iam::111122223333:role/JohnDoe"
      ]
    },
    "Effect": "Allow",
    "Action": ["s3:GetObject", "s3:GetObjectVersion"],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {
      "StringEquals":
        {"s3:ExistingObjectTag/environment": "production"}
    }
  }
  ]
}
```

Example 2: Restrict which object tag keys that users can add

The following permissions policy grants a user permissions to perform the `s3:PutObjectTagging` action, which allows user to add tags to an existing object. The condition

uses the `s3:RequestObjectTagKeys` condition key to specify the allowed tag keys, such as `Owner` or `CreationDate`. For more information, see Creating a condition that tests multiple key values in the *IAM User Guide*.

The policy ensures that every tag key specified in the request is an authorized tag key. The `ForAnyValue` qualifier in the condition ensures that at least one of the specified keys must be present in the request.

JSON

```
{
   "Version":"2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
            "arn:aws:iam::111122223333:role/JohnDoe"
         ]
       },
 "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
  ]
}
```

Example 3: Require a specific tag key and value when allowing users to add object tags

The following example policy grants a user permission to perform the `s3:PutObjectTagging` action, which allows a user to add tags to an existing object. The condition requires the user to include a specific tag key (such as *`Project`*) with the value set to *`X`*.

JSON

```
{
   "Version":"2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
         ]
       },
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

Managing object tags

This section explains how you can manage object tags using the AWS SDKs for Java and .NET or the Amazon S3 console.

Object tagging gives you a way to categorize storage in general purpose buckets. Each tag is a key-value pair that adheres to the following rules:

- You can associate up to 10 tags with an object. Tags that are associated with an object must have unique tag keys.
- A tag key can be up to 128 Unicode characters in length, and tag values can be up to 256 Unicode characters in length. Amazon S3 object tags are internally represented in UTF-16. Note that in UTF-16, characters consume either 1 or 2 character positions.
- The key and values are case sensitive.

For more information about object tags, see Tagging your objects. For more information about tag restrictions, see User-Defined Tag Restrictions in the *AWS Billing and Cost Management User Guide*.

Using the S3 console

To add tags to an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the bucket list, choose the name of the bucket that contains the object.
4. Select the check box to the left of the names of the objects you want to change.
5. In the **Actions** menu, choose **Edit tags**.
6. Review the objects listed, and choose **Add tags**.
7. Each object tag is a key-value pair. Enter a **Key** and a **Value**. To add another tag, choose **Add Tag**.

   You can enter up to 10 tags for an object.
8. Choose **Save changes**.

   Amazon S3 adds the tags to the specified objects.

For more information, see also Viewing object properties in the Amazon S3 console and Uploading objects in this guide.

Using the AWS SDKs

Java

To manage object tags using the AWS SDK for Java, you can set tags for a new object and retrieve or replace tags for an existing object. For more information about object tagging, see Tagging your objects.

Upload an object to a bucket and set tags using an S3Client. For examples, see Upload an object to a bucket in the *Amazon S3 API Reference*.

.NET

The following example shows how to use the AWS SDK for .NET to set the tags for a new object and retrieve or replace the tags for an existing object. For more information about object tagging, see Tagging your objects.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    public class ObjectTagsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for the new object ***";
        private const string filePath = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            PutObjectWithTagsTestAsync().Wait();
        }

        static async Task PutObjectWithTagsTestAsync()
        {
            try
            {
                // 1. Put an object with tags.
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
```

```
                    FilePath = filePath,
                    TagSet = new List<Tag>{
                        new Tag { Key = "Keyx1", Value = "Value1"},
                        new Tag { Key = "Keyx2", Value = "Value2" }
                    }
                };

                PutObjectResponse response = await
client.PutObjectAsync(putRequest);
                // 2. Retrieve the object's tags.
                GetObjectTaggingRequest getTagsRequest = new GetObjectTaggingRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);
                for (int i = 0; i < objectTags.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}",
objectTags.Tagging[i].Key, objectTags.Tagging[i].Value);


                // 3. Replace the tagset.

                Tagging newTagSet = new Tagging();
                newTagSet.TagSet = new List<Tag>{
                    new Tag { Key = "Key3", Value = "Value3"},
                    new Tag { Key = "Key4", Value = "Value4" }
                };


                PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    Tagging = newTagSet
                };
                PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

                // 4. Retrieve the object's tags.
```

```
                GetObjectTaggingRequest getTagsRequest2 = new
 GetObjectTaggingRequest();
                getTagsRequest2.BucketName = bucketName;
                getTagsRequest2.Key = keyName;
                GetObjectTaggingResponse objectTags2 = await
 client.GetObjectTaggingAsync(getTagsRequest2);
                for (int i = 0; i < objectTags2.Tagging.Count; i++)
                    Console.WriteLine("Key: {0}, Value: {1}",
 objectTags2.Tagging[i].Key, objectTags2.Tagging[i].Value);

            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an
 object"
                        , e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Encountered an error. Message:'{0}' when writing an object"
                    , e.Message);
            }
        }
    }
}
```

Editing object metadata in the Amazon S3 console

You can use the Amazon S3 console to edit metadata for existing S3 objects by using the **Copy** action. To edit metadata, you copy objects to the same destination and specify the new metadata you want to apply, which replaces the old metadata for the object. Some metadata is set by Amazon S3 when you upload the object. For example, `Content-Length` and `Last-Modified` are system-defined object metadata fields that can't be modified by a user.

You can also set user-defined metadata when you upload the object and replace it as your needs change. For example, you might have a set of objects that you initially store in the `STANDARD` storage class. Over time, you may no longer need this data to be highly available. So, you can change the storage class to `GLACIER` by replacing the value of the `x-amz-storage-class` key from `STANDARD` to `GLACIER`.

Note

Consider the following when you are replacing system-defined or user-defined object metadata using the **Copy** action:

- You must specify existing metadata you want to retain, metadata you want to add, and metadata you want to edit.
- If your object is less than 5 GB, you can use the **Copy** action in the S3 console to replace object metadata. If your object is greater than 5 GB, you can replace the object metadata when you copy an object with multipart upload by using the AWS CLI or AWS SDKs. For more information, see Copying an object using multipart upload.
- For a list of additional permissions required to replace metadata, see the section called "Required permissions for S3 API operations". For example policies that grant this permission, see the section called "Identity-based policy examples".
- This action creates a *copy* of the object with updated settings and the last-modified date. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. If S3 Versioning isn't enabled, a new copy of the object replaces the original object. The AWS account associated with the IAM role that changes the property also becomes the owner of the new object or (object version).
- Editing metadata replaces values for existing key names.
- Objects that are encrypted with customer-provided encryption keys (SSE-C) can't be copied by using the console. You must use the AWS CLI, AWS SDK, or the Amazon S3 REST API.
- When copying an object by using the Amazon S3 console, you might receive the error message "Copied metadata can't be verified." The console uses headers to retrieve and set metadata for your object. If your network or browser configuration modifies your network requests, this behavior might cause unintended metadata (such as modified `Cache-Control` headers) to be written to your copied object. Amazon S3 can't verify this unintended metadata.

  To address this issue, check your network and browser configuration to make sure it doesn't modify headers, such as `Cache-Control`. For more information, see The Shared Responsibility Model.

Warning

When replacing metadata for folders, wait for the **Copy** action to finish before adding new objects to the folder. Otherwise, new objects might also be edited.

The following topics describe how to replace metadata for an object by using the **Copy** action in the Amazon S3 console.

Note

To add, view, or delete annotations on an object, use the **Annotations** section on the object's **Properties** tab. For more information, see Managing annotations.

Replacing system-defined metadata

You can replace some system-defined metadata for an S3 object. For a list of system-defined metadata and values that you can modify, see System-defined object metadata.

To replace system-defined metadata of an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets** or **Directory buckets**.
3. In the list of buckets, choose the name of the bucket that contains the objects you want to change.
4. Select the check box for the objects you want to change.
5. On the **Actions** menu, choose **Copy** from the list of options that appears.
6. To specify the destination path, choose **Browse S3**, navigate to the same destination as the source objects, and select the destination check box. Choose **Choose destination**.

   Alternatively, enter the destination path.
7. If you do *not* have bucket versioning enabled, you will see a warning recommending you enable Bucket Versioning to help protect against unintentionally overwriting or deleting objects. If you want to keep all versions of objects in this bucket, select **Enable Bucket**

Versioning. You can also view the default encryption and Object Lock properties in **Destination details**.

8. Under **Additional copy settings**, choose **Specify settings** to specify settings for **Metadata**.
9. Scroll to the **Metadata** section, and then choose **Replace all metadata**.
10. Choose **Add metadata**.
11. For metadata **Type**, select **System-defined**.
12. Specify a unique **Key** and the metadata **Value**.
13. To edit additional metadata, choose **Add metadata**. You can also choose **Remove** to remove a set of type-key-values.
14. Choose **Copy**. Amazon S3 saves your metadata changes.

Replacing user-defined metadata

You can replace user-defined metadata of an object by combining the metadata prefix, `x-amz-meta-`, and a name you choose to create a custom key. For example, if you add the custom name `alt-name`, the metadata key would be `x-amz-meta-alt-name`.

User-defined metadata can be as large as 2 KB total. To calculate the total size of user-defined metadata, sum the number of bytes in the UTF-8 encoding for each key and value. Both keys and their values must conform to US-ASCII standards. For more information, see User-defined metadata.

To replace user-defined metadata of an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Buckets**, and then choose the **General purpose buckets** or **Directory buckets** tab. Navigate to the Amazon S3 bucket or folder that contains the objects you want to change.
3. Select the check box for the objects you want to change.
4. On the **Actions** menu, choose **Copy** from the list of options that appears.
5. To specify the destination path, choose **Browse S3**, navigate to the same destination as the source objects, and select the destination check box. Choose **Choose destination**.

   Alternatively, enter the destination path.

6. If you do *not* have bucket versioning enabled, you will see a warning recommending you enable Bucket Versioning to help protect against unintentionally overwriting or deleting objects. If you want to keep all versions of objects in this bucket, select **Enable Bucket Versioning**. You can also view the default encryption and Object Lock properties in **Destination details**.
7. Under **Additional copy settings**, choose **Specify settings** to specify settings for **Metadata**.
8. Scroll to the **Metadata** section, and then choose **Replace all metadata**.
9. Choose **Add metadata**.
10. For metadata **Type**, choose **User-defined**.
11. Enter a unique custom **Key** following `x-amz-meta-`. Also enter a metadata **Value**.
12. To add additional metadata, choose **Add metadata**. You can also choose **Remove** to remove a set of type-key-values.
13. Choose **Copy**. Amazon S3 saves your metadata changes.

Discovering your data with S3 Metadata tables

Amazon S3 Metadata accelerates data discovery by automatically capturing metadata for objects in your general purpose buckets and storing it in read-only, fully managed Apache Iceberg tables that you can query. These read-only tables are called *metadata tables*. As objects are added to, updated, or removed from your general purpose buckets, S3 Metadata automatically refreshes the corresponding metadata tables to reflect the latest changes.

By default, S3 Metadata provides three types of metadata:

- System-defined metadata, such as an object's creation time and storage class
- Custom object metadata, such as tags, annotations, and user-defined metadata that was included during object upload
- Event metadata, such as when an object is updated or deleted, and the AWS account that made the request

With S3 Metadata, you can easily find, store, and query metadata for your S3 objects, so that you can quickly prepare data for use in business analytics, content retrieval, artificial intelligence and machine learning (AI/ML) model training, and more.

For each general purpose bucket, you can create a metadata table configuration that contains up to three metadata tables:

- **Journal table** – By default, your metadata table configuration contains a *journal table*, which captures events that occur for the objects in your bucket. The journal table records changes made to your data in near real time, helping you to identify new data uploaded to your bucket, track recently deleted objects, monitor lifecycle transitions, and more. The journal table records new objects and updates to your objects and their metadata (those updates that require either a `PUT` or a `DELETE` operation).

  The journal table captures metadata only for change events (such as uploads, updates, and deletes) that happen after you create your metadata table configuration. Because this table is queryable, you can audit the changes to your bucket through simple SQL queries.

  The journal table is required for each metadata table configuration. (In the initial release of S3 Metadata, the journal table was referred to as "the metadata table.")

  For more information about what data is stored in journal tables, see the section called “Journal tables schema”.

  To help minimize your storage costs, you can choose to enable journal table record expiration. For more information, see the section called “Expiring journal table records”.

- **Live inventory table** – Optionally, you can add a *live inventory table* to your metadata table configuration. The live inventory table provides a simple, queryable inventory of all the objects and their versions in your bucket so that you can determine the latest state of your data.

  You can use the live inventory table to simplify and speed up business workflows and big data jobs by identifying objects that you want to process for various workloads. For example, you can query the live inventory table to find all objects stored in a particular storage class, all objects with certain tags, all objects that aren't encrypted with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS), and more.

  When you enable the live inventory table for your metadata table configuration, the table goes through a process known as *backfilling*, during which Amazon S3 scans your general purpose bucket to retrieve the initial metadata for all objects that exist in the bucket. Depending on the number of objects in your bucket, this process can take minutes (minimum 15 minutes) to hours. When the backfilling process is finished, the status of your live inventory table changes from **Backfilling** to **Active**. After backfilling is completed, updates to your objects are typically reflected in the live inventory table within one hour.

You're charged for backfilling your inventory table. If your general purpose bucket has more than one billion objects, you're also charged a monthly fee for your live inventory table. For more information, see Amazon S3 Pricing.

For more information about what data is stored in live inventory tables, see the section called "Live inventory tables schema".

- **Annotation table** – Optionally, you can add an *annotation table* to your metadata table configuration. The annotation table tracks the latest annotations on the objects in your bucket and makes annotation content directly queryable. Each row represents one annotation on a specific object version.

  You can use the annotation table to discover and retrieve objects based on annotation values. For example, you can query the annotation table to find all objects with a specific annotation key, retrieve annotation content that matches a predicate based on object size or key prefix, or correlate annotations across objects.

  For more information about configuring the annotation table, see the section called "Enabling or disabling annotation tables". For more information about the annotation table schema, see the section called "Annotation table schema".

Your metadata tables are stored in an AWS managed S3 table bucket, which provides storage that's optimized for tabular data. To query your metadata, you can integrate your table bucket with Amazon SageMaker Lakehouse. This integration, which uses the AWS Glue Data Catalog and AWS Lake Formation, allows AWS analytics services to automatically discover and access your table data.

After your table bucket is integrated with the AWS Glue Data Catalog, you can directly query your metadata tables with AWS analytics services such as Amazon Athena, Amazon EMR, and Amazon Redshift. You can also create interactive dashboards with your query data by using AWS Quick. For more information about integrating your AWS managed S3 table bucket with Amazon SageMaker Lakehouse, see the section called "Integrating S3 Tables with AWS analytics services".

You can also query your metadata tables with Apache Spark, Apache Trino, and any other application that supports the Apache Iceberg format by using the AWS Glue Iceberg REST endpoint, the Amazon S3 Tables Iceberg REST endpoint, or the Amazon S3 Tables Catalog for Apache Iceberg client catalog. For more information about accessing your metadata tables, see the section called "Accessing tables".

For S3 Metadata pricing, see Amazon S3 Pricing.

How metadata tables work

Metadata tables are managed by Amazon S3, and can't be modified by any IAM principal outside of Amazon S3 itself. You can, however, delete your metadata tables. As a result, metadata tables are read-only, which helps ensure that they correctly reflect the contents of your general purpose bucket.

To generate and store object metadata in AWS managed metadata tables, you create a metadata table configuration for your general purpose bucket. Amazon S3 is designed to continuously update the metadata tables to reflect the latest changes to your data as long as the configuration is active on the general purpose bucket.

Before you create a metadata table configuration, make sure that you have the necessary AWS Identity and Access Management (IAM) permissions to create and manage metadata tables. For more information, see the section called “Permissions for metadata tables”.

Metadata table storage, organization, and encryption

When you create your metadata table configuration, your metadata tables are stored in an AWS managed table bucket. All metadata table configurations in your account and in the same Region are stored in a single AWS managed table bucket. These AWS managed table buckets are named `aws-s3` and have the following Amazon Resource Name (ARN) format:

`arn:aws:s3tables:`*`region`*`:`*`account_id`*`:bucket/aws-s3`

For example, if your account ID is 123456789012 and your general purpose bucket is in US East (N. Virginia) (`us-east-1`), your AWS managed table bucket is also created in US East (N. Virginia) (`us-east-1`) and has the following ARN:

`arn:aws:s3tables:`*`us-east-1`*`:123456789012:bucket/aws-s3`

By default, AWS managed table buckets are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3). After you create your first metadata configuration, you can set the default encryption setting for the AWS managed table bucket to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS). For more information, see Encryption for AWS managed table buckets and the section called “Specifying SSE-KMS”.

Within your AWS managed table bucket, the metadata tables for your configuration are typically stored in a namespace with the following naming format:

b_*general-purpose-bucket-name*

Note

- If your general purpose bucket name contains any periods, the periods are converted to underscores (_) in the namespace name.
- If your general purpose bucket was created before March 1, 2018, its name might contain uppercase letters and underscores, and it might also be up to 255 characters long. If your bucket name has these characteristics, your metadata table namespace will have a different format. The general purpose bucket name will be prefixed with b_, truncated to 63 characters, converted to all lowercase, and suffixed with a hash.

Metadata tables have the following Amazon Resource Name (ARN) format, which includes the table ID of the metadata table:

`arn:aws:s3tables:`*`region-code`*`:`*`account-id`*`:bucket/aws-s3/table/`*`table-id`*

For example, a metadata table in the US East (N. Virginia) Region would have an ARN like the following:

```
arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3/table/
a12bc345-67d8-912e-3456-7f89123g4h56
```

Journal tables have the name `journal`, and live inventory tables have the name `inventory`.

When you create your metadata table configuration, you can choose to encrypt your AWS managed metadata tables with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS). If you choose to use SSE-KMS, you must provide a customer managed KMS key in the same Region as your general purpose bucket. You can set the encryption type for your tables only during table creation. After an AWS managed table is created, you can't change its encryption setting. To specify SSE-KMS for your metadata tables, you must have certain permissions. For more information, see Permissions for SSE-KMS.

The encryption setting for a metadata table takes precedence over the default bucket-level encryption setting. If you don't specify encryption for a table, it will inherit the default encryption setting from the bucket.

AWS managed table buckets don't count toward your S3 Tables quotas. For more information about working with AWS managed table buckets and AWS managed tables, see Working with AWS managed table buckets.

To monitor updates to your metadata table configuration, you can use AWS CloudTrail. For more information, see the section called "Amazon S3 bucket-level actions that are tracked by CloudTrail logging".

Metadata table maintenance and record expiration

To keep your metadata tables performing at their best, Amazon S3 performs periodic maintenance activities on your tables, such as compaction and unreferenced file removal. These maintenance activities help to both minimize the cost of storing your metadata tables and optimize query performance. This table maintenance happens automatically, requiring no opt-in or ongoing management by you.

> **Note**
>
> - You can't control the expiration of journal table or inventory table snapshots. For each table, Amazon S3 stores a minimum of 1 snapshot for a maximum of 24 hours.
> - To help minimize your costs, you can configure journal table record expiration. By default, journal table records don't expire, and journal table records must be retained for a minimum of 7 days. For more information, see the section called "Expiring journal table records".

Topics

- Metadata table limitations and restrictions
- S3 Metadata journal tables schema
- S3 Metadata live inventory tables schema
- S3 Metadata annotation table schema
- Configuring metadata tables
- Querying metadata tables
- Troubleshooting S3 Metadata

Metadata table limitations and restrictions

Amazon S3 Metadata has the following limitations and restrictions:

- S3 Metadata is currently available only in certain Regions. For more information, see the section called "S3 Metadata AWS Regions".
- S3 Metadata supports all storage classes supported by general purpose buckets. For the S3 Intelligent-Tiering storage class, the specific tier isn't shown in the metadata table.
- When you create a metadata table configuration, your metadata tables are stored in an AWS managed table bucket. You can't store your configuration in a customer-managed table bucket.
- S3 Metadata isn't supported for directory buckets, table buckets, or vector buckets. You can create metadata table configurations only for general purpose buckets. The journal table captures metadata only for change events (such as uploads, updates, and deletes) that happen after you have created your metadata table configuration.
- You can't control the expiration of journal table or inventory table snapshots. For each table, Amazon S3 stores a minimum of 1 snapshot for a maximum of 24 hours.

  To help minimize your costs, you can configure journal table record expiration. By default, journal table records don't expire, and journal table records must be retained for a minimum of 7 days. For more information, see the section called "Expiring journal table records".
- You can create a metadata table configuration only for an entire general purpose bucket. You can't apply a metadata table configuration at the prefix level.
- You can't pause and resume updates to metadata tables. However, you can delete your associated metadata configuration for journal or live inventory tables. Deleting your configuration doesn't delete the associated journal or inventory table. To re-create your configuration, you must first delete the old journal or inventory table, and then Amazon S3 can create a new journal or inventory table. When you re-enable the inventory table, Amazon S3 creates a new inventory table, and you're charged again for backfilling the new inventory table.
- Metadata tables don't contain all the same metadata as is available through S3 Inventory or through the Amazon S3 REST API. For example, the following information isn't available in metadata tables:
  - S3 Lifecycle expiration eligibility or transition status
  - Object Lock retention period or governance mode
  - Object access control list (ACL) information
  - Replication status
- When you're using Amazon Athena or Amazon Redshift to query your metadata tables, you must surround your metadata table namespace names in quotation marks (") or backticks (`),

otherwise the query might not work. For examples, see the section called "Example metadata table queries".

- When using Apache Spark on Amazon EMR or other third-party engines to query your metadata tables, we recommend that you use the Amazon S3 Tables Iceberg REST endpoint. Your query might not run successfully if you don't use this endpoint. For more information, see the section called "Accessing tables using the Amazon S3 Tables Iceberg REST endpoint".

S3 Metadata AWS Regions

S3 Metadata is currently available in the following AWS Regions:

AWS Region name	AWS Region code
Africa (Cape Town)	af-south-1
Asia Pacific (Hong Kong)	ap-east-1
Asia Pacific (Hyderabad)	ap-south-2
Asia Pacific (Jakarta)	ap-southeast-3
Asia Pacific (Malaysia)	ap-southeast-5
Asia Pacific (Melbourne)	ap-southeast-4
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (New Zealand)	ap-southeast-6
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Taipei)	ap-east-2
Asia Pacific (Thailand)	ap-southeast-7

AWS Region name	AWS Region code
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
Canada West (Calgary)	ca-west-1
Europe (Frankfurt)	eu-central-1
Europe (Zurich)	eu-central-2
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Milan)	eu-south-1
Europe (Paris)	eu-west-3
Europe (Spain)	eu-south-2
Europe (Stockholm)	eu-north-1
Israel (Tel Aviv)	il-central-1
Mexico (Central)	mx-central-1
Middle East (Bahrain)	me-south-1
Middle East (UAE)	me-central-1
South America (São Paulo)	sa-east-1
US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2

AWS Region name	AWS Region code
China (Beijing)	`cn-north-1`
China (Ningxia)	`cn-northwest-1`
AWS GovCloud (US-East)	`us-gov-east-1`
AWS GovCloud (US-West)	`us-gov-west-1`

S3 Metadata journal tables schema

The journal table records changes made to your data in near real time, helping you to identify new data uploaded to your bucket, track recently deleted objects, monitor lifecycle transitions, and more. The journal table records new objects and updates to your objects and their metadata (those updates that require either a PUT or a DELETE operation). Because this table is queryable, you can audit the changes to your bucket through simple SQL queries.

You can use the journal table for security, auditing, and compliance use cases to track uploaded, deleted, and changed objects in the bucket. For example, you can query the journal table to answer questions such as:

- Which objects were deleted in the past 24 hours by S3 Lifecycle?
- Which IP addresses did the most recent PUT requests come from?
- Which AWS Key Management Service (AWS KMS) keys were used for PUT requests in the past 7 days?
- Which objects in your bucket were created by Amazon Bedrock in the last five days?

Amazon S3 Metadata journal tables contain rows and columns. Each row represents a mutation event that has created, updated, or deleted an object in your general purpose bucket. Most of these events result from user actions, but some of these events result from actions taken by Amazon S3 on your behalf, such as S3 Lifecycle expirations or storage class transitions.

S3 Metadata journal tables are eventually consistent with the changes that have occurred in your general purpose bucket. In some cases, by the time S3 Metadata is notified that an object is created or updated, that object might already have been overwritten or deleted in the bucket. In such

cases, the objects can no longer be retrieved and some columns might show a NULL value to indicate missing metadata schema.

The following is an example of a journal table for a general purpose bucket named `amzn-s3-demo-bucket`:

```
bucket               key                         sequence_number
                                                                    record_type
 record_timestamp          version_id   is_delete_marker   size   last_modified_date
   e_tag                            storage_class  is_multipart   encryption_status
 is_bucket_key_enabled   kms_key_arn
                checksum_algorithm   object_tags   user_metadata

    requester      source_ip_address   request_id
amzn-s3-demo-bucket   Finance/statement1.pdf
 80e737d8b4d82f776affffffffffffff006737d8b4d82f776a00000000000000000000000000000000000000000000
   CREATE        2024-11-15 23:26:44.899                 FALSE
 6223   11/15/2024 23:26     e131b86632dda753aac4018f72192b83    STANDARD
 FALSE          SSE-KMS             FALSE                   arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890df   SSECRC32             {}
            {count -> Asia, customs -> false, family -> true, location -> Mary, name ->
 football, user -> United States}                       111122223333   192.0.2.1
    CVK8FWYRW0M9JW65
amzn-s3-demo-bucket   s3-dg.pdf
 80e737d8b4e39f1dbdffffffffffffff006737d8b4e39f1dbd00000000000000000000000000000000000000000000
   CREATE        2024-11-15 23:26:44.942                 FALSE
 3554   11/15/2024 23:26     9bb49efc2d92c05558ddffbbde8636d5    STANDARD
 FALSE          DSSE-KMS            FALSE                   arn:aws:kms:us-
east-1:936810216292:key/0dcebce6-49fd-4cae-b2e2-5512ad281afd   SSESHA1              {}
           {}
                                                  111122223333   192.0.2.1
   CVKAQDRAZEG7KXAY
amzn-s3-demo-bucket   Development/Projects.xls
 80e737d8b4ed9ac5c6ffffffffffffff006737d8b4ed9ac5c600000000000000000000000000000000000000000000
   CREATE        2024-11-15 23:26:44.966                 FALSE              7746
 11/15/2024 23:26     729a6863e47fb9955b31bfabce984908    STANDARD   FALSE
 SSE-S3              FALSE                   NULL
                                  SSECRC32             {}            {count -> Asia,
 customs -> Canada, family -> Billiards, filter -> true, location -> Europe, name ->
 Asia, user -> United States}   111122223333   192.0.2.1           CVK7Z6XQTQ90BSRV
```

Journal tables have the following schema:

Column name	Required?	Data type	
`bucket`	Yes	String	The general purpose bucket name. For more information, see the section called "Naming rules".
`key`	Yes	String	The object key name (or key) that uniquely identifies the object in the bucket. For more information, see the section called "Naming objects".
`sequence_number`	Yes	String	The sequence number, which is an ordinal that's included in the records for a given object. To order records of the same bucket and key, you can sort on `sequence_number` . For a given bucket and key, a lexicographically larger `sequence_number` value implies that the record was introduced to the bucket more recently.
`record_type`	Yes	String	The type of this record, one of

Column name	Required?	Data type	
			CREATE, UPDATE_METADATA , DELETE, CREATE_ANNOTATION , DELETE_ANNOTATION , or UPDATE_ANNOTATION_METADATA . CREATE records indicate that a new object (or a new version of the object) was written to the bucket. UPDATE_METADATA records capture changes to mutable metadata for an existing object, such as the storage class or tags. DELETE records indicate that this object (or this version of the object) has been deleted. When versioning is enabled, DELETE records represent either a delete marker or a permanent delete. They are further

Column name	Required?	Data type	
			disambiguated by consulting the optional `is_delete_marker` column. `CREATE_ANNOTATION` records introduce a new annotation on an object, or an overwrite of an existing annotation. `DELETE_ANNOTATION` records indicate that an annotation has been deleted. `UPDATE_ANNOTATION_METADATA` records capture changes to mutable metadata for an existing annotation, such as when the replication status changes. For more information, see the section called "Deleting object versions".

Column name	Required?	Data type	
			Note A permanent delete carries NULLs in all columns, *except* `bucket`, `key`, `sequence_number`, `record_type`, `record_timestamp`, and `version_id` (i.e. those columns marked as Required).
`record_timestamp`	Yes	Timestamp NTZ (no time zone)	The timestamp that's associated with this record.

Column name	Required?	Data type	
`version_id`	No	String	The object's version ID. When you enable versioning on a bucket, Amazon S3 assigns a version number to objects that are added to the bucket. For more information, see the section called "Retaining multiple versions of objects". Objects that are stored in your bucket before you set the versioning state have a version ID of null.

Column name	Required?	Data type	
`is_delete_marker`	No	Boolean	The object's delete marker status. For DELETE records that are delete markers, this value is TRUE. For permanent deletions, this value is omitted (NULL). Other record types (CREATE and UPDATE_METADATA) have value FALSE. For more information, see the section called "Working with delete markers". **Note** Rows that are added for delete markers have a `record_type` value of DELETE, not `UPDATE_METADATA`. If the delete marker is created as the result of an S3 Lifecycle expiration, the `requester`

Column name	Required?	Data type	
			value is `s3.amazonaws.com` .
`size`	No	Long	The object size in bytes, not including the size of incomplete multipart uploads or object metadata. If `is_delete_marker` is TRUE, the size is `0`. For more information, see the section called "System-defined object metadata".
`last_modified_date`	No	Timestamp NTZ (no time zone)	The object creation date or the last modified date, whichever is the latest. For multipart uploads, the object creation date is the date when the multipart upload is initiated. For more information, see the section called "System-defined object metadata".

Column name	Required?	Data type	
e_tag	No	String	The entity tag (ETag), which is a hash of the object. The ETag reflects changes only to the contents of an object, not to its metadata. The ETag can be an MD5 digest of the object data. Whether the ETag is an MD5 digest depends on how the object was created and how it's encrypted. For more information, see Object in the *Amazon S3 API Reference*.

Column name	Required?	Data type	
storage_class	No	String	The storage class that's used for storing the object. One of STANDARD, REDUCED_REDUNDANCY, STANDARD_IA, ONEZONE_IA, INTELLIGENT_TIERING, GLACIER, DEEP_ARCHIVE, or GLACIER_IR. For more information, see the section called "Understanding and managing storage classes".
is_multipart	No	Boolean	The object's upload type. If the object was uploaded as a multipart upload, this value is TRUE. Otherwise, it's FALSE. For more information, see the section called "Using multipart upload".

Column name	Required?	Data type	
`encryption_status`	No	String	The object's server-side encryption status, depending on what kind of encryption key is used: server-side encryption with Amazon S3 managed keys (SSE-S3), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If the object is unencrypted, this value is null. Possible values are `SSE-S3`, `SSE-KMS`, `DSSE-KMS`, `SSE-C`, or null. For more information, see the section called “Data encryption”.

Column name	Required?	Data type	
`is_bucket_key_enabled`	No	Boolean	The object's S3 Bucket Key enablement status. If the object uses an S3 Bucket Key for SSE-KMS, this value is TRUE. Otherwise, it's FALSE. For more information, see the section called "Configuring an S3 Bucket Key for an object".

Column name	Required?	Data type	
kms_key_arn	No	String	The Amazon Resource Name (ARN) for the KMS key with which the object is encrypted, for rows where encryption_status is SSE-KMS or DSSE-KMS. If the object isn't encrypted with SSE-KMS or DSSE-KMS, the value is null. For more information, see the section called "KMS keys stored in AWS KMS (SSE-KMS)" and the section called "Dual-layer server-side encryption (DSSE-KMS)". **Note** If a row represents an object version that no longer existed at the time that a delete or overwrite event was processed, kms_key_a

Column name	Required?	Data type	
			rn contains a null value, even if the encryptio n_status column value is SSE-KMS or DSSE-KMS.
checksum_ algorithm	No	String	The algorithm that's used to create the checksum for the object, one of CRC64NVME , CRC32, CRC32C, SHA1, SHA256, MD5, XXHASH64, XXHASH3, XXHASH128 , or SHA512. If no checksum is present, this value is null. For more information, see the section called "Using supported checksum algorithm s".

Column name	Required?	Data type	
`object_tags`	No	Map <String, String>	The object tags that are associated with the object. Object tags are stored as a map of key-value pairs. If an object has no object tags, an empty map ({}) is stored. For more information, see the section called "Tagging objects". **Note** If the `record_type` value is `DELETE`, the `object_tags` column contains a null value. If the `record_type` value is `CREATE` or `UPDATE_METADATA`, rows that represent object versions that no longer existed at

Column name	Required?	Data type	
			the time that a delete or overwrite event was processed will contain a null value in the `object_tags` column.

Column name	Required?	Data type	
`user_metadata`	No	Map <String, String>	The user metadata that's associated with the object. User metadata is stored as a map of key-value pairs. If an object has no user metadata, an empty map ({}) is stored. For more information, see the section called "User-defined metadata". **Note** If the `record_type` value is `DELETE`, the `user_metadata` column contains a null value. If the `record_type` value is `CREATE` or `UPDATE_METADATA`, rows that represent object versions that no longer

Column name	Required?	Data type	
			existed at the time that a delete or overwrite event was processed will contain a null value in the `user_metadata` column.
`requester`	No	String	The AWS account ID of the requester or the AWS service principal that made the request. For example, if the requester is S3 Lifecycle, this value is `s3.amazonaws.com`.

Column name	Required?	Data type	
source_ip_address	No	String	The source IP address of the request. For records that are generated by a user request, this column contains the source IP address of the request. For actions taken by Amazon S3 or another AWS service on behalf of the user, this column contains a null value.
request_id	No	String	The request ID that's associated with the request.
object_join_key	No	String	A unique identifier for this object, assigned with each new version, or when the null version is created or overwritten. Use this column to join with the annotation table.

Column name	Required?	Data type	
annotation.name	No	String	The name of the annotation being modified. Present only for CREATE_ANNOTATION, UPDATE_ANNOTATION_METADATA, and DELETE_ANNOTATION record types.
annotation.sequence_number	No	String	An ordinal value for sorting annotation revisions.
annotation.last_modified_date	No	Timestamp	The date and time when the annotation was last modified.
annotation.size	No	Long	The size of the annotation payload in bytes.
annotation.e_tag	No	String	The ETag of the annotation content.

Column name	Required?	Data type	
annotation.replication_status	No	String	The replication status of the annotation. PENDING, COMPLETED , and FAILED are visible only at the replication source. REPLICA is visible only at the replication destination. If there is no applicable replication configuration, this value is NULL.
annotation.checksum_algorithm	No	String	The checksum algorithm used for the annotation.

S3 Metadata live inventory tables schema

The live inventory table provides a simple, queryable inventory of all the objects and their versions in your bucket so that you can determine the latest state of your data. Updates to your objects are typically reflected in the inventory table within one hour.

You can use this table to simplify and speed up business workflows and big data jobs by identifying objects that you want to process for various workloads. For example, you can query the inventory table to do the following:

- Find all objects stored in the S3 Glacier Deep Archive storage class.
- Create a distribution of object tags or find objects without tags.
- Find all objects that aren't encrypted by using server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS).

When you enable the inventory table for your metadata table configuration, the table goes through a process known as *backfilling*, during which Amazon S3 scans your general purpose bucket to retrieve the initial metadata for all objects in the bucket. Depending on the number of objects in your bucket, this process can take minutes (minimum 15 minutes) to hours. When the backfilling process is finished, the status of your inventory table changes from **Backfilling** to **Active**. After backfilling is completed, updates to your objects are typically reflected in the inventory table within one hour.

> **Note**
>
> You're charged for backfilling your inventory table. If your general purpose bucket has more than one billion objects, you're also charged a monthly fee for your inventory table. For more information, see Amazon S3 Pricing.

Amazon S3 Metadata inventory tables contain rows and columns. Each row represents the current state of an object in your general purpose bucket. The inventory table provides a simple, queryable inventory of all objects in your bucket so that you can determine the current state of your data.

The following is an example of an inventory table for a general purpose bucket named `amzn-s3-demo-bucket`:

```
bucket               key                         sequence_number
                                                                    version_id
   is_delete_marker   size   last_modified_date   e_tag
 storage_class   is_multipart   encryption_status   is_bucket_key_enabled   kms_key_arn
                                                              checksum_algorithm
 object_tags   user_metadata

amzn-s3-demo-bucket   Finance/statement1.pdf
 80e737d8b4d82f776affffffffffffff006737d8b4d82f776a00000000000000000000000000000000000000000000
               FALSE              6223   11/15/2024 23:26
  e131b86632dda753aac4018f72192b83    STANDARD   FALSE
   SSE-KMS             FALSE                   arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890df   SSECRC32             {}
           {count -> Asia, customs -> false, family -> true, location -> Mary, name ->
 football, user -> United States}
amzn-s3-demo-bucket   s3-dg.pdf
 80e737d8b4e39f1dbdffffffffffffff006737d8b4e39f1dbd00000000000000000000000000000000000000000000
               FALSE              3554   11/15/2024 23:26
 9bb49efc2d92c05558ddffbbde8636d5    STANDARD   FALSE          DSSE-KMS
```

```
 FALSE                   arn:aws:kms:us-east-1:936810216292:key/0dcebce6-49fd-4cae-
b2e2-5512ad281afd   SSESHA1             {}            {}


amzn-s3-demo-bucket   Development/Projects.xls
 80e737d8b4ed9ac5c6fffffffffffffff006737d8b4ed9ac5c6000000000000000000000000000000000000000000000
                FALSE              7746   11/15/2024 23:26
 729a6863e47fb9955b31bfabce984908    STANDARD   FALSE          SSE-S3
 FALSE                   NULL
                SSECRC32             {}            {count -> Asia, customs -> Canada,
 family -> Billiards, filter -> true, location -> Europe, name -> Asia, user -> United
 States}
```

Inventory tables have the following schema:

Column name	Required?	Data type	
`bucket`	Yes	String	The general purpose bucket name. For more information, see the section called "Naming rules".
`key`	Yes	String	The object key name (or key) that uniquely identifies the object in the bucket. For more information, see the section called "Naming objects".
`sequence_number`	Yes	String	The sequence number, which is an ordinal that's included in the records for a given object. To order records of the same bucket and key,

Column name	Required?	Data type	
			you can sort on `sequence_number`. For a given bucket and key, a lexicographically larger `sequence_number` value implies that the record was introduced to the bucket more recently.
`object_join_key`	No	String	A unique identifier for the object, assigned with each new version, or when the null version is created or overwritten. Use this column to join with the annotation table.

Column name	Required?	Data type	
`version_id`	No	String	The object's version ID. When you enable versioning on a bucket, Amazon S3 assigns a version number to objects that are added to the bucket. For more information, see the section called "Retaining multiple versions of objects". Objects that are stored in your bucket before you set the versioning state have a version ID of null.

Column name	Required?	Data type	
`is_delete_marker`	No	Boolean	The object's delete marker status. If the object is a delete marker, this value is `True`. Otherwise, it's `False`. For more information, see the section called "Working with delete markers". **Note** Rows that are added for delete markers have a `record_type` value of `DELETE`, not `UPDATE_METADATA`. If the delete marker is created as the result of an S3 Lifecycle expiration, the `requester` value is `s3.amazonaws.com`.

Column name	Required?	Data type	
`size`	No	Long	The object size in bytes, not including the size of incomplete multipart uploads or object metadata. If `is_delete_marker` is `True`, the size is `0`. For more information, see the section called "System-defined object metadata".
`last_modified_date`	No	Timestamp NTZ (no time zone)	The object creation date or the last modified date, whichever is the latest. For multipart uploads, the object creation date is the date when the multipart upload is initiated. For more information, see the section called "System-defined object metadata".

Column name	Required?	Data type	
e_tag	No	String	The entity tag (ETag), which is a hash of the object. The ETag reflects changes only to the contents of an object, not to its metadata. The ETag can be an MD5 digest of the object data. Whether the ETag is an MD5 digest depends on how the object was created and how it's encrypted. For more information, see Object in the *Amazon S3 API Reference*.

Column name	Required?	Data type	
storage_class	No	String	The storage class that's used for storing the object. One of STANDARD, REDUCED_REDUNDANCY, STANDARD_IA, ONEZONE_IA, INTELLIGENT_TIERING, GLACIER, DEEP_ARCHIVE, or GLACIER_IR. For more information, see the section called "Understanding and managing storage classes".
is_multipart	No	Boolean	The object's upload type. If the object was uploaded as a multipart upload, this value is True. Otherwise, it's False. For more information, see the section called "Using multipart upload".

Column name	Required?	Data type	
`encryption_status`	No	String	The object's server-side encryption status, depending on what kind of encryption key is used: server-side encryption with Amazon S3 managed keys (SSE-S3), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If the object is unencrypted, this value is null. Possible values are `SSE-S3`, `SSE-KMS`, `DSSE-KMS`, `SSE-C`, or null. For more information, see the section called “Data encryption”.

Column name	Required?	Data type	
`is_bucket_key_enabled`	No	Boolean	The object's S3 Bucket Key enablement status. If the object uses an S3 Bucket Key for SSE-KMS, this value is `True`. Otherwise, it's `False`. For more information, see the section called "Configuring an S3 Bucket Key for an object".

Column name	Required?	Data type	
kms_key_arn	No	String	The Amazon Resource Name (ARN) for the KMS key with which the object is encrypted, for rows where encryption_status is SSE-KMS or DSSE-KMS. If the object isn't encrypted with SSE-KMS or DSSE-KMS, the value is null. For more information, see the section called "KMS keys stored in AWS KMS (SSE-KMS)" and the section called "Dual-layer server-side encryption (DSSE-KMS)". **Note** If a row represents an object version that no longer existed at the time that a delete or overwrite event was processed, kms_key_a

Column name	Required?	Data type	
			`rn` contains a null value, even if the `encryptio n_status` column value is `SSE-KMS` or `DSSE-KMS`.
`checksum_ algorithm`	No	String	The algorithm that's used to create the checksum for the object, one of `CRC64-NVME` , `CRC32`, `CRC32C`, `SHA1`, `SHA256`, `MD5`, `XXHASH64`, `XXHASH3`, `XXHASH128` , or `SHA512`. If no checksum is present, this value is null. For more information, see the section called "Using supported checksum algorithm s".

Column name	Required?	Data type	
object_tags	No	Map <String, String>	The object tags that are associated with the object. Object tags are stored as a map of key-value pairs. If an object has no object tags, an empty map ({}) is stored. For more information, see the section called "Tagging objects". **Note** If the record_type value is DELETE, the object_tags column contains a null value. If the record_type value is CREATE or UPDATE_METADATA, rows that represent object versions that no longer existed at

Column name	Required?	Data type	
			the time that a delete or overwrite event was processed will contain a null value in the `object_tags` column.

Column name	Required?	Data type	
`user_metadata`	No	Map <String, String>	The user metadata that's associated with the object. User metadata is stored as a map of key-value pairs. If an object has no user metadata, an empty map ({}) is stored. For more information, see the section called "User-defined metadata". **Note** If the `record_type` value is `DELETE`, the `user_metadata` column contains a null value. If the `record_type` value is `CREATE` or `UPDATE_METADATA`, rows that represent object versions that no longer

Column name	Required?	Data type	
			existed at the time that a delete or overwrite event was processed will contain a null value in the `user_metadata` column.

S3 Metadata annotation table schema

The annotation table tracks the latest annotations on the objects in your general purpose bucket. Each row represents one annotation on a specific object version. You can use the annotation table to discover and retrieve objects based on annotation content. For example, you can query the annotation table to find all objects with a specific annotation, retrieve annotation content that matches a predicate based on object size or key prefix, or correlate annotations across objects.

When you enable the annotation table for your metadata table configuration, the table goes through a process known as *backfilling*, during which Amazon S3 scans your general purpose bucket to retrieve the annotation data for all objects. Depending on the number of objects in your bucket, this process can take minutes to hours. When backfilling is finished, the status changes from **Backfilling** to **Active**.

> **Note**
>
> You're charged for backfilling your annotation table. For more information, see Amazon S3 Pricing.

Annotation tables have the following schema:

Column name	Required?	Data type	Description
`bucket`	Yes	String	The name of the general purpose bucket.
`object_key`	Yes	String	The object key name (or key) that uniquely identifies the object in the bucket.
`object_version_id`	No	String	The object's version ID.
`object_join_key`	Yes	String	A unique identifier for the object, assigned with each new version, or when the null version is created or overwritten. Use this column to join with the inventory or journal tables.
`name`	Yes	String	The annotation name.
`sequence_number`	Yes	String	An ordinal value for sorting annotation revisions. For a given bucket, `object_key`, and `object_version_id`, a lexicographically larger `sequence_number` implies the

Column name	Required?	Data type	Description
			record was introduced more recently.
`last_modified_date`	No	Timestamp NTZ (no time zone)	The date and time when the annotation was last modified.
`size`	No	Long	The size of the annotation payload in bytes.
`e_tag`	No	String	The ETag of the annotation content.
`checksum_algorithm`	No	String	The checksum algorithm used for the annotation. If no checksum is present, this value is null.
`replication_status`	No	String	The replication status of the annotation. `PENDING`, `COMPLETED` , and `FAILED` are visible only at the replication source. `REPLICA` is visible only at the replication destination. If there is no applicable replication configuration, this value is null.
`text_value`	No	String	The annotation content (UTF-8 text).

Configuring metadata tables

Amazon S3 Metadata accelerates data discovery by automatically capturing metadata for the objects in your general purpose buckets and storing it in read-only, fully managed Apache Iceberg tables that you can query. These read-only tables are called *metadata tables*. As objects are added to, updated, and removed from your general purpose buckets, S3 Metadata automatically refreshes the corresponding metadata tables to reflect the latest changes.

With S3 Metadata, you can easily find, store, and query metadata for your S3 objects, so that you can quickly prepare data for use in business analytics, artificial intelligence and machine learning (AI/ML) model training, and more.

To generate and store object metadata in AWS managed metadata tables, you create a metadata table configuration for your general purpose bucket. Amazon S3 is designed to continuously update the metadata tables to reflect the latest changes to your data as long as the configuration is active on the bucket. Additionally, Amazon S3 continuously optimizes your metadata tables to help reduce storage costs and improve analytics query performance.

To create a metadata table configuration, make sure that you have the necessary AWS Identity and Access Management (IAM) permissions to create and manage metadata tables.

To monitor updates to your metadata table configuration, you can use AWS CloudTrail. For more information, see the section called “Amazon S3 bucket-level actions that are tracked by CloudTrail logging”.

Topics

- Setting up permissions for configuring metadata tables
- Creating metadata table configurations
- Controlling access to metadata tables
- Expiring journal table records
- Enabling or disabling live inventory tables
- Enabling or disabling annotation tables
- Viewing metadata table configurations
- Deleting metadata table configurations
- Deleting metadata tables

Setting up permissions for configuring metadata tables

To create a metadata table configuration, you must have the necessary AWS Identity and Access Management (IAM) permissions to both create and manage your metadata table configuration and to create and manage your metadata tables and the table bucket where your metadata tables are stored.

To create and manage your metadata table configuration, you must have these permissions:

- `s3:CreateBucketMetadataTableConfiguration` – This permission allows you to create a metadata table configuration for your general purpose bucket. To create a metadata table configuration, additional permissions, including S3 Tables permissions, are required, as explained in the following sections. For a summary of the required permissions, see the section called "Bucket operations and permissions".
- `s3:GetBucketMetadataTableConfiguration` – This permission allows you to retrieve information about your metadata table configuration.
- `s3:DeleteBucketMetadataTableConfiguration` – This permission allows you to delete your metadata table configuration.
- `s3:UpdateBucketMetadataJournalTableConfiguration` – This permission allows you to update your journal table configuration to expire journal table records.
- `s3:UpdateBucketMetadataInventoryTableConfiguration` – This permission allows you to update your inventory table configuration to enable or disable the inventory table. To update an inventory table configuration, additional permissions, including S3 Tables permissions, are required. For a list of the required permissions, see the section called "Bucket operations and permissions".
- `s3:UpdateBucketMetadataAnnotationTableConfiguration` – This permission allows you to update your annotation table configuration to enable or disable the annotation table. To update an annotation table configuration, additional permissions, including S3 Tables permissions, are required. For a list of the required permissions, see the section called "Bucket operations and permissions".

> **Note**
>
> The `s3:CreateBucketMetadataTableConfiguration`, `s3:GetBucketMetadataTableConfiguration`, and `s3:DeleteBucketMetadataTableConfiguration` permissions are used for both V1 and V2 S3 Metadata configurations.

For V2, the names of the corresponding API operations are `CreateBucketMetadataConfiguration`, `GetBucketMetadataConfiguration`, and `DeleteBucketMetadataConfiguration`.

- `iam:PassRole` – To create an Amazon S3 Metadata configuration that includes an annotation table, you need this permission to pass the annotation table IAM role. Add the following statement to your caller's IAM policy:

```
{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "arn:aws:iam::123456789012:role/annotation-table-role",
    "Condition": {"StringEquals": {"iam:PassedToService":
 "metadata.s3.amazonaws.com"}}
}
```

To create and work with tables and table buckets, you must have certain `s3tables` permissions. At a minimum, to create a metadata table configuration, you must have the following `s3tables` permissions:

- `s3tables:CreateTableBucket` – This permission allows you to create an AWS managed table bucket. All metadata table configurations in your account and in the same Region are stored in a single AWS managed table bucket named `aws-s3`. For more information, see the section called "How metadata tables work" and Working with AWS managed table buckets.
- `s3tables:CreateNamespace` – This permission allows you to create a namespace in a table bucket. Metadata tables typically use the b_*general_purpose_bucket_name* namespace. For more information about metadata table namespaces, see the section called "How metadata tables work".
- `s3tables:CreateTable` – This permission allows you to create your metadata tables.
- `s3tables:GetTable` – This permission allows you to retrieve information about your metadata tables.
- `s3tables:PutTablePolicy` – This permission allows you to add or update your metadata table policies.
- `s3tables:PutTableEncryption` – This permission allows you to set server-side encryption for your metadata tables. Additional permissions are required if you want to encrypt your

metadata tables with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS). For more information, see Permissions for SSE-KMS.

- `kms:DescribeKey` – This permission allows you to retrieve information about a KMS key.
- `s3tables:PutTableBucketPolicy` – This permission allows you to create or update a new table bucket policy.

For detailed information about all table and table bucket permissions, see Access management for S3 Tables.

> **Important**
>
> If you also want to integrate your table bucket with AWS analytics services so that you can query your metadata table, you need additional permissions. For more information, see Integrating Amazon S3 Tables with AWS analytics services.

Permissions for SSE-KMS

To encrypt your metadata tables with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), you must have additional permissions.

1. The user or AWS Identity and Access Management (IAM) role needs the following permissions. You can grant these permissions by using the IAM console: https://console.aws.amazon.com/iam/.
   a. `s3tables:PutTableEncryption` to configure table encryption
   b. `kms:DescribeKey` on the AWS KMS key used
2. On the resource policy for the KMS key, you need the following permissions. You can grant these permissions by using the AWS KMS console: https://console.aws.amazon.com/kms.
   a. Grant `kms:GenerateDataKey` permission to `metadata.s3.amazonaws.com` and `maintenance.s3tables.amazonaws.com`.
   b. Grant `kms:Decrypt` permission to `metadata.s3.amazonaws.com` and `maintenance.s3tables.amazonaws.com`.
   c. Grant `kms:DescribeKey` permission to the invoking AWS principal.

In addition to these permissions, make sure that the customer managed KMS key used to encrypt the tables still exists, is active, is in the same Region as your general purpose bucket.

Example policy

To create and work with metadata tables and table buckets, you can use the following example policy. In this policy, the general purpose bucket that you're applying the metadata table configuration to is referred to as *amzn-s3-demo-bucket*. To use this policy, replace the *user input placeholders* with your own information.

When you create your metadata table configuration, your metadata tables are stored in an AWS managed table bucket. All metadata table configurations in your account and in the same Region are stored in a single AWS managed table bucket named `aws-s3`.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionsToWorkWithMetadataTables",
            "Effect": "Allow",
            "Action": [
                "s3:CreateBucketMetadataTableConfiguration",
                "s3:GetBucketMetadataTableConfiguration",
                "s3:DeleteBucketMetadataTableConfiguration",
                "s3:UpdateBucketMetadataJournalTableConfiguration",
                "s3:UpdateBucketMetadataInventoryTableConfiguration",
                "s3:UpdateBucketMetadataAnnotationTableConfiguration",
                "s3tables:*",
                "kms:DescribeKey"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3",
                "arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3/table/*",
                "arn:aws:kms:Region:Account:key/KmsKeyId"
            ]
        }
    ]
}
```

To query metadata tables, you can use the following example policy. If your metadata tables have been encrypted with SSE-KMS, you will need the `kms:Decrypt` permission as shown. To use this policy, replace the *`user input placeholders`* with your own information.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionsToQueryMetadataTables",
            "Effect": "Allow",
            "Action": [
                "s3tables:GetTable",
                "s3tables:GetTableData",
                "s3tables:GetTableMetadataLocation",
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3",
                "arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3/table/*",
                "arn:aws:kms:Region:Account:key/KmsKeyId"
            ]
        }
    ]
}
```

Creating metadata table configurations

To generate and store Amazon S3 Metadata in fully managed Apache Iceberg metadata tables, you create a metadata table configuration for your general purpose bucket. Amazon S3 is designed to continuously update the metadata tables to reflect the latest changes to your data as long as the configuration is active on the bucket. Additionally, Amazon S3 continuously optimizes your metadata tables to help reduce storage costs and improve analytics query performance.

For each general purpose bucket, you can create a metadata table configuration that contains up to three complementary metadata tables:

- **Journal table** – By default, your metadata table configuration contains a *journal table*, which captures events that occur for the objects in your bucket. The journal table records changes made to your data in near real time, helping you to identify new data uploaded to your bucket, track recently deleted objects, monitor lifecycle transitions, and more. The journal table records

new objects and updates to your objects and their metadata (those updates that require either a PUT or a DELETE operation).

The journal table captures metadata only for change events (such as uploads, updates, and deletes) that happen after you create your metadata table configuration. Because this table is queryable, you can audit the changes to your bucket through simple SQL queries.

The journal table is required for each metadata table configuration. (In the initial release of S3 Metadata, the journal table was referred to as "the metadata table.")

For more information about what data is stored in journal tables, see the section called “Journal tables schema”.

To help minimize your storage costs, you can choose to enable journal table record expiration. For more information, see the section called “Expiring journal table records”.

- **Live inventory table** – Optionally, you can add a *live inventory table* to your metadata table configuration. The live inventory table provides a simple, queryable inventory of all the objects and their versions in your bucket so that you can determine the latest state of your data.

  You can use the live inventory table to simplify and speed up business workflows and big data jobs by identifying objects that you want to process for various workloads. For example, you can query the live inventory table to find all objects stored in a particular storage class, all objects with certain tags, all objects that aren't encrypted with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS), and more.

  When you enable the live inventory table for your metadata table configuration, the table goes through a process known as *backfilling*, during which Amazon S3 scans your general purpose bucket to retrieve the initial metadata for all objects that exist in the bucket. Depending on the number of objects in your bucket, this process can take minutes (minimum 15 minutes) to hours. When the backfilling process is finished, the status of your live inventory table changes from **Backfilling** to **Active**. After backfilling is completed, updates to your objects are typically reflected in the live inventory table within one hour.

  You're charged for backfilling your live inventory table. If your general purpose bucket has more than one billion objects, you're also charged a monthly fee for your live inventory table. For more information, see Amazon S3 Pricing.

  For more information about what data is stored in live inventory tables, see the section called “Live inventory tables schema”.

- **Annotation table** – Optionally, you can add an *annotation table* to your metadata table configuration. The annotation table tracks the latest annotations on the objects in your bucket and makes annotation content directly queryable. The annotation table is not enabled by default. For more information, see the section called "Enabling or disabling annotation tables".

Metadata tables have the following Amazon Resource Name (ARN) format, which includes the table ID of the metadata table:

`arn:aws:s3tables:region-code:account-id:bucket/aws-s3/table/table-id`

For example, a metadata table in the US East (N. Virginia) Region would have an ARN like the following:

```
arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3/table/
a12bc345-67d8-912e-3456-7f89123g4h56
```

Journal tables have the name `journal`, live inventory tables have the name `inventory`, and annotation tables have the name `annotation`.

When you create your metadata table configuration, your metadata tables are stored in an AWS managed table bucket. All metadata table configurations in your account and in the same Region are stored in a single AWS managed table bucket. These AWS managed table buckets are named `aws-s3` and have the following Amazon Resource Name (ARN) format:

`arn:aws:s3tables:region:account_id:bucket/aws-s3`

For example, if your account ID is 123456789012 and your general purpose bucket is in US East (N. Virginia) (`us-east-1`), your AWS managed table bucket is also created in US East (N. Virginia) (`us-east-1`) and has the following ARN:

`arn:aws:s3tables:us-east-1:123456789012:bucket/aws-s3`

By default, AWS managed table buckets are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3). After you create your first metadata configuration, you can set the default encryption setting for the AWS managed table bucket to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS). For more information, see Encryption for AWS managed table buckets and the section called "Specifying SSE-KMS".

Within your AWS managed table bucket, the metadata tables for your configuration are typically stored in a namespace with the following naming format:

b_*general-purpose-bucket-name*

For more information about metadata table namespaces, see the section called "How metadata tables work".

When you create your metadata table configuration, you can choose to encrypt your AWS managed metadata tables with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS). If you choose to use SSE-KMS, you must provide a customer managed KMS key in the same Region as your general purpose bucket. You can set the encryption type for your tables only during table creation. After an AWS managed table is created, you can't change its encryption setting. To specify SSE-KMS for your metadata tables, you must have certain permissions. For more information, see Permissions for SSE-KMS.

The encryption setting for a metadata table takes precedence over the default bucket-level encryption setting. If you don't specify encryption for a table, it will inherit the default encryption setting from the bucket.

AWS managed table buckets don't count toward your S3 Tables quotas. For more information about working with AWS managed table buckets and AWS managed tables, see Working with AWS managed table buckets.

You can create a metadata table configuration by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

Note

- If you created your S3 Metadata configuration before July 15, 2025, we recommend that you delete and re-create your configuration so that you can expire journal table records, create an inventory table, and create an annotation table. For more information, see the section called "Enabling inventory and annotation tables on metadata configurations created before July 15, 2025".
- If you've deleted your metadata table configuration and want to re-create a configuration for the same general purpose bucket, you must first manually delete the old journal, inventory, and annotation tables from your AWS managed table bucket. Otherwise, creating the new metadata table configuration fails because those tables already exist. To delete your metadata tables, see the section called "Delete a metadata table".

Deleting a metadata table configuration deletes only the configuration. The AWS managed table bucket and your metadata tables still exist, even if you delete the metadata table configuration.

Prerequisites

Before you create a metadata table configuration make sure that you've met the following prerequisites:

- Before you create a metadata table configuration make sure that you have the necessary AWS Identity and Access Management (IAM) permissions to create and manage metadata tables. For more information, see the section called "Permissions for metadata tables".
- If you plan to query your metadata tables with Amazon Athena or another AWS query engine, make sure that you integrate your AWS managed table bucket with AWS analytics services. For more information, see the section called "Integrating S3 Tables with AWS analytics services".

  If you've already integrated an existing table bucket in this Region, your AWS managed table bucket is also automatically integrated. To determine the integration status for your table buckets in this Region, open the Amazon S3 console, and choose **Table buckets** in the left navigation pane. Under **Integration with AWS analytics services**, check the Region and whether the integration status says **Enabled**.

Create a metadata table configuration

Using the S3 console

To create a metadata table configuration

Before you create a metadata table configuration, make sure that you've reviewed and met the prerequisites and that you've reviewed the section called "Limitations and restrictions".

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. Choose the general purpose bucket that you want to create a metadata table configuration for.

> **Note**
>
> Make sure that this general purpose bucket is an AWS Region where table buckets are available. Table buckets are available only in the US East (N. Virginia), US East (Ohio), and US West (Oregon) Regions.

4. On the bucket's details page, choose the **Metadata** tab.
5. On the **Metadata** tab, choose **Create metadata configuration**.
6. On the **Create metadata configuration** page, under **Journal table**, you can choose whether to encrypt your table with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS). By default, journal tables are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3).

   If you choose to use SSE-KMS, you must provide a customer managed KMS key in the same Region as your general purpose bucket.

   > **Important**
   >
   > You can set the encryption type for your metadata tables only during table creation. After an AWS managed table is created, you can't change its encryption setting.

   - To encrypt your journal table with SSE-S3 (the default), choose **Don't specify encryption type**.
   - To encrypt your journal table with SSE-KMS, choose **Specify encryption type**. Under **Encryption type**, choose **Server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS)**. Under **AWS KMS key**, either choose from your existing KMS keys, or enter your KMS key ARN. If you don't already have a KMS key, choose **Enter KMS key ARN**, and then choose **Create a KMS key**.

     Make sure that you've set up the necessary permissions for SSE-KMS. For more information, see Permissions for SSE-KMS.
7. (Optional) By default, the records in your journal table don't expire. To help minimize the storage costs for your journal table, choose **Enabled** for **Record expiration**.

If you enable journal table record expiration, you can set the number of days to retain your journal table records. To set the **Days after which records expire** value, you can specify any whole number between 7 and 2147483647. For example, to retain your journal table records for one year, set this value to 365.

Records will be expired within 24 to 48 hours after they become eligible for expiration.

> **Important**
>
> After journal table records expire, they can't be recovered.

Under **Journal table records will expire after the specified number of days**, select the checkbox.

8. (Optional) If you want to add an inventory table to your metadata table configuration, under **Live inventory table**, choose **Enabled** for **Configuration status**.

   You can choose whether to encrypt your table with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS). By default, inventory tables are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3).

   If you choose to use SSE-KMS, you must provide a customer managed KMS key in the same Region as your general purpose bucket.

   > **Important**
   >
   > You can set the encryption type for your metadata tables only during table creation. After an AWS managed table is created, you can't change its encryption setting.

   - To encrypt your inventory table with SSE-S3 (the default), choose **Don't specify encryption type**.
   - To encrypt your inventory table with SSE-KMS, choose **Specify encryption type**. Under **Encryption type**, choose **Server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS)**. Under **AWS KMS key**, either choose from your existing KMS keys, or enter your KMS key ARN. If you don't already have a KMS key, choose **Enter KMS key ARN**, and then choose **Create a KMS key**.

Make sure that you've set up the necessary permissions for SSE-KMS. For more information, see Permissions for SSE-KMS.

9. (Optional) If you want to add an annotation table to your metadata table configuration, under **Annotation table**, choose **Enabled** for **Configuration status**.

   Under **IAM role**, choose one of the following IAM role selection methods:

   - **Create new IAM role** – Amazon S3 creates a new IAM role with the required permissions for the annotation table.
   - **Choose from existing IAM roles** – Select an existing IAM role from the **IAM role** dropdown. The role must have the required permissions to read annotations from your bucket.
   - **Enter IAM role ARN** – Enter the ARN of an existing IAM role that has the required permissions.

   For more information about the required permissions for the annotation table IAM role, see Annotation table IAM role.

   You can choose whether to encrypt your annotation table with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS). By default, annotation tables are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3).

   If you choose to use SSE-KMS, you must provide a customer managed KMS key in the same Region as your general purpose bucket.

   > **Important**
   >
   > You can set the encryption type for your metadata tables only during table creation. After an AWS managed table is created, you can't change its encryption setting.

   - To encrypt your annotation table with SSE-S3 (the default), choose **Don't specify encryption type**.
   - To encrypt your annotation table with SSE-KMS, choose **Specify encryption type**. Under **Encryption type**, choose **Server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS)**. Under **AWS KMS key**, either choose from your existing KMS

keys, or enter your KMS key ARN. If you don't already have a KMS key, choose **Enter KMS key ARN**, and then choose **Create a KMS key**.

Make sure that you've set up the necessary permissions for SSE-KMS. For more information, see Permissions for SSE-KMS.

10. Choose **Create metadata table configuration**.

If your metadata table configuration was successful, the names and ARNs for your metadata tables are displayed on the **Metadata** tab, along with the name of your AWS managed table bucket and namespace.

If you chose to enable an inventory table for your metadata table configuration, the table goes through a process known as *backfilling*, during which Amazon S3 scans your general purpose bucket to retrieve the initial metadata for all objects that exist in the bucket. Depending on the number of objects in your bucket, this process can take minutes (minimum 15 minutes) to hours. When the backfilling process is finished, the status of your inventory table changes from **Backfilling** to **Active**. After backfilling is completed, updates to your objects are typically reflected in the inventory table within one hour.

To monitor updates to your metadata table configuration, you can use AWS CloudTrail. For more information, see the section called “Amazon S3 bucket-level actions that are tracked by CloudTrail logging”.

Using the AWS CLI

To run the following commands, you must have the AWS CLI installed and configured. If you don't have the AWS CLI installed, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

Alternatively, you can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? and Getting started with AWS CloudShell in the *AWS CloudShell User Guide*.

To create a metadata table configuration by using the AWS CLI

Before you create a metadata table configuration, make sure that you've reviewed and met the prerequisites and that you've reviewed the section called “Limitations and restrictions”.

To use the following example commands, replace the *user input placeholders* with your own information.

1. Create a JSON file that contains your metadata table configuration, and save it (for example, `metadata-config.json`). The following is a sample configuration.

   You must specify whether to enable or disable journal table record expiration. If you choose to enable record expiration, you must also specify the number of days after which your journal table records will expire. To set the `Days` value, you can specify any whole number between 7 and 2147483647. For example, to retain your journal table records for one year, set this value to 365.

   You can optionally choose to configure an inventory table and an annotation table.

   For journal tables, inventory tables, and annotation tables, you can optionally specify an encryption configuration. By default, metadata tables are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3), which you can specify by setting `SseAlgorithm` to `AES256`.

   To encrypt your metadata tables with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS), set `SseAlgorithm` to `aws:kms`. You must also set `KmsKeyArn` to the ARN of a customer managed KMS key in the same Region where your general purpose bucket is located.

```
{
  "JournalTableConfiguration": {
     "RecordExpiration": {
       "Expiration": "ENABLED",
       "Days": 10
    },
    "EncryptionConfiguration": {
      "SseAlgorithm": "AES256"
    }
  },
  "InventoryTableConfiguration": {
    "ConfigurationState": "ENABLED",
    "EncryptionConfiguration": {
      "SseAlgorithm": "aws:kms",
      "KmsKeyArn": "arn:aws:kms:us-east-2:account-id:key/key-id"
    }
  },
```

```
  "AnnotationTableConfiguration": {
    "ConfigurationState": "ENABLED",
    "Role": "arn:aws:iam::account-id:role/annotation-table-role",
    "EncryptionConfiguration": {
      "SseAlgorithm": "AES256"
    }
  }
}
```

2. Use the following command to apply the metadata table configuration to your general purpose bucket (for example, *amzn-s3-demo-bucket*):

```
aws s3api create-bucket-metadata-configuration \
--bucket amzn-s3-demo-bucket \
--metadata-configuration file://./metadata-config.json \
--region us-east-2
```

3. To verify that the configuration was created, use the following command:

```
aws s3api get-bucket-metadata-configuration \
--bucket amzn-s3-demo-bucket \
--region us-east-2
```

To monitor updates to your metadata table configuration, you can use AWS CloudTrail. For more information, see the section called "Amazon S3 bucket-level actions that are tracked by CloudTrail logging".

Using the REST API

You can send REST requests to create a metadata table configuration. For more information, see CreateBucketMetadataConfiguration in the *Amazon S3 API Reference*.

Using the AWS SDKs

You can use the AWS SDKs to create a metadata table configuration in Amazon S3. For information, see the list of supported SDKs in the *Amazon S3 API Reference*.

Enabling inventory and annotation tables on metadata configurations created before July 15, 2025

If you created your S3 Metadata configuration before July 15, 2025, we recommend that you delete and re-create your configuration so that you can expire journal table records, create an

inventory table, and create an annotation table. Any changes to your general purpose bucket that occur between deleting the old configuration and creating the new one aren't recorded in either of your journal tables.

To migrate from an old metadata configuration to a new configuration, do the following:

1. Delete your existing metadata table configuration. For step-by-step instructions, see the section called “Deleting metadata table configurations”.
2. Create a new metadata table configuration. For step-by-step instructions, see the section called “Creating metadata tables”.

If you need assistance with migrating your configuration, contact AWS Support.

After you create your new metadata configuration, you will have two journal tables. If you no longer need the old journal table, you can delete it. For step-by-step instructions, see the section called “Deleting metadata tables”. If you've retained your old journal table and want to join it with your new one, see the section called “Joining custom metadata” for examples of how to join two tables.

After migration, you can do the following:

1. To view your configuration, you can now use the `GetBucketMetadataConfiguration` API operation. To determine whether your configuration is old or new, you can look at the following attribute of your `GetBucketMetadataConfiguration` API response. An AWS managed bucket type (`"aws"`) indicates a new configuration, and a customer-managed bucket type (`"customer"`) indicates an old configuration.

   ```
   "MetadataTableConfigurationResult": {
              "TableBucketType": ["aws" | "customer"]
   ```

   For more information, see the section called “Viewing metadata table configurations”.

   > **Note**
   >
   > You can use the `GetBucketMetadataConfiguration` and `DeleteBucketMetadataConfiguration` API operations with old or new metadata table configurations. However, if you try to use the `GetBucketMetadataTableConfiguration` and

`DeleteBucketMetadataTableConfiguration` API operations with new configurations, you will receive HTTP `405 Method Not Allowed` errors. Make sure that you update your processes to use the new API operations (`CreateBucketMetadataConfiguration`, `GetBucketMetadataConfiguration`, and `DeleteBucketMetadataConfiguration`) instead of the old API operations.

2. If you plan to query your metadata tables with Amazon Athena or another AWS query engine, make sure that you integrate your AWS managed table bucket with AWS analytics services. If you've already integrated an existing table bucket in this Region, your AWS managed table bucket is also automatically integrated. For more information, see the section called "Integrating S3 Tables with AWS analytics services".

Controlling access to metadata tables

To control access to your Amazon S3 metadata tables, you can use AWS Identity and Access Management (IAM) resource-based policies that are attached to your table bucket and to your metadata tables. In other words, you can control access to your metadata tables at both the table bucket level and the table level.

For more information about controlling access to your table buckets and tables, see Access management for S3 Tables.

Important

When you're creating or updating table bucket or table policies, make sure that you don't restrict the Amazon S3 service principals `metadata.s3.amazonaws.com` and `maintenance.s3tables.amazonaws.com` from writing to your table bucket or your metadata tables.
If Amazon S3 is unable to write to your table bucket or your metadata tables, you must delete your metadata configuration, delete your metadata tables, and then create a new configuration. If you had an inventory table in your configuration, a new inventory table has to be created, and you will be charged again for backfilling the new inventory table.

You can also control access to the rows and columns in your metadata tables through AWS Lake Formation. For more information, see Managing Lake Formation permissions and Data filtering and cell-level security in Lake Formation in the *AWS Lake Formation Developer Guide*.

Expiring journal table records

By default, the records in your journal table don't expire. To help minimize the storage costs for your journal table, you can enable journal table record expiration.

> **Note**
>
> If you created your S3 Metadata configuration before July 15, 2025, you can't enable journal table record expiration on that configuration. We recommend that you delete and re-create your configuration so that you can expire journal table records and create an inventory table. For more information, see the section called "Enabling inventory and annotation tables on metadata configurations created before July 15, 2025".

If you enable journal table record expiration, you can set the number of days to retain your journal table records. To set this value, specify any whole number between 7 and 2147483647. For example, to retain your journal table records for one year, set this value to 365.

> **Important**
>
> After journal table records expire, they can't be recovered.

Records are expired within 24 to 48 hours after they become eligible for expiration. Journal records are removed from the latest snapshot. The data and storage for the deleted records is removed through table maintenance operations.

If you've enabled journal table record expiration, you can disable it at any time to stop expiring your journal table records.

You can expire journal table records by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

How to expire journal table records

Using the S3 console

To expire journal table records

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **General purpose buckets**.
3. Choose the general purpose bucket that contains the metadata table configuration with the journal table that you want to expire records from.
4. On the bucket's details page, choose the **Metadata** tab.
5. On the **Metadata** tab, choose **Edit**, then choose **Edit journal table record expiration**.
6. On the **Edit journal table record expiration** page, choose **Enabled** under **Record expiration**.
7. Set the number of days to retain your journal table records. To set the **Days after which records expire** value, specify any whole number between 7 and 2147483647. For example, to retain your journal table records for one year, set this value to 365.

   > **Important**
   >
   > After journal table records expire, they can't be recovered.

8. Under **Journal table records will expire after the specified number of days**, select the checkbox.
9. Choose **Save changes**.

If you want to disable journal table record expiration, repeat the preceding steps, but choose **Disabled** instead of **Enabled** for step 6.

Using the AWS CLI

To run the following commands, you must have the AWS CLI installed and configured. If you don't have the AWS CLI installed, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

You can also run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? and Getting started with AWS CloudShell in the *AWS CloudShell User Guide*.

To expire journal table records by using the AWS CLI

To use the following example commands, replace the *`user input placeholders`* with your own information.

1. Create a JSON file that contains your journal table configuration, and save it (for example, `journal-config.json`). The following is a sample configuration.

   To set the `Days` value, specify any whole number between 7 and 2147483647. For example, to retain your journal table records for one year, set this value to 365.

   ```
   {
     "RecordExpiration": {
       "Expiration": "ENABLED",
       "Days": 10
     }
   }
   ```

   To disable journal table record expiration, create the following sample configuration instead. If `Expiration` is set to `DISABLED`, you must not specify a `Days` value in the configuration.

   ```
   {
     "RecordExpiration": {
       "Expiration": "DISABLED"
     }
   }
   ```

2. Use the following command to expire records from the journal table in your general purpose bucket (for example, *amzn-s3-demo-bucket*):

   ```
   aws s3api update-bucket-metadata-journal-table-configuration \
   --bucket amzn-s3-demo-bucket \
   --journal-table-configuration file://./journal-config.json \
   --region us-east-2
   ```

Using the REST API

You can send REST requests to expire journal table records. For more information, see UpdateBucketMetadataJournalTableConfiguration.

Using the AWS SDKs

You can use the AWS SDKs to expire journal table records in Amazon S3. For information, see the list of supported SDKs.

Enabling or disabling live inventory tables

By default, your metadata table configuration contains a *journal table*, which records the events that occur for the objects in your bucket. The journal table is required for each metadata table configuration.

Optionally, you can add a *live inventory table* to your metadata table configuration. The live inventory table provides a simple, queryable inventory of all the objects and their versions in your bucket so that you can determine the latest state of your data.

> **Note**
>
> If you created your S3 Metadata configuration before July 15, 2025, you can't enable an inventory table on that configuration. We recommend that you delete and re-create your configuration so that you can create an inventory table and expire journal table records. For more information, see the section called “Enabling inventory and annotation tables on metadata configurations created before July 15, 2025”.

The inventory table contains the latest metadata for all objects in your bucket. You can use this table to simplify and speed up business workflows and big data jobs by identifying objects that you want to process for various workloads. For example, you can query the inventory table to do the following:

- Find all objects stored in the S3 Glacier Deep Archive storage class.
- Create a distribution of object tags or find objects without tags.
- Find all objects that aren't encrypted by using server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS).
- Compare your inventory table at two different points in time to understand the growth in objects with specific tags.

If you chose to enable an inventory table for your metadata table configuration, the table goes through a process known as *backfilling*, during which Amazon S3 scans your general purpose bucket to retrieve the initial metadata for all objects that exist in the bucket. Depending on the number of objects in your bucket, this process can take minutes (minimum 15 minutes) to hours. When the backfilling process is finished, the status of your inventory table changes from

Backfilling to **Active**. After backfilling is completed, updates to your objects are typically reflected in the inventory table within one hour.

> **Note**
>
> - You're charged for backfilling your inventory table. If your general purpose bucket has more than one billion objects, you're also charged a monthly fee for your inventory table. For more information, see Amazon S3 Pricing.
> - You can't pause updates to your inventory table and then resume them. However, you can disable the inventory table configuration. Disabling the inventory table doesn't delete it. The inventory table is retained for your records until you decide to delete it.
>
> If you've disabled your inventory table and later want to re-enable it, you must first delete the old inventory table from your AWS managed table bucket. When you re-enable the inventory table configuration, Amazon S3 creates a new inventory table, and you're charged again for backfilling the new inventory table.

You can enable or disable inventory tables by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

Prerequisites

If you've disabled your inventory table and now want to re-enable it, you must first manually delete the old inventory table from your AWS managed table bucket. Otherwise, re-enabling the inventory table fails because an inventory table already exists in the table bucket. To delete your inventory table, see the section called “Delete a metadata table”.

When you re-enable the inventory table configuration, Amazon S3 creates a new inventory table, and you're charged again for backfilling the new inventory table.

Enable or disable inventory tables

Using the S3 console

To enable or disable inventory tables

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **General purpose buckets**.
3. Choose the general purpose bucket with the metadata table configuration that you want to enable or disable an inventory table for.
4. On the bucket's details page, choose the **Metadata** tab.
5. On the **Metadata** tab, choose **Edit**, then choose **Edit inventory table configuration**.
6. On the **Edit inventory table configuration** page, choose **Enabled** or **Disabled** under **Inventory table**.

> **Note**
>
> Before you choose **Enabled**, make sure that you've reviewed and met the prerequisites.

- If you chose **Enabled**, you can choose whether to encrypt your table with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS). By default, inventory tables are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3).

  If you choose to use SSE-KMS, you must provide a customer managed KMS key in the same Region as your general purpose bucket.

  > **Important**
  >
  > You can set the encryption type for your metadata tables only during table creation. After an AWS managed table is created, you can't change its encryption setting.

  - To encrypt your inventory table with SSE-S3 (the default), choose **Don't specify encryption type**.
  - To encrypt your inventory table with SSE-KMS, choose **Specify encryption type**. Under **Encryption type**, choose **Server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS)**. Under **AWS KMS key**, either choose from your existing KMS keys, or enter your KMS key ARN. If you don't already have a KMS key, choose **Enter KMS key ARN**, and then choose **Create a KMS key**.
- If you chose **Disabled**, under **After the inventory table is disabled, the table will no longer be updated, and updates can't be resumed**, select the checkbox.

7. Choose **Save changes**.

Using the AWS CLI

To run the following commands, you must have the AWS CLI installed and configured. If you don't have the AWS CLI installed, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

Alternatively, you can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? and Getting started with AWS CloudShell in the *AWS CloudShell User Guide*.

To enable or disable inventory tables by using the AWS CLI

To use the following example commands, replace the *`user input placeholders`* with your own information.

> **Note**
>
> Before enabling an inventory configuration, make sure that you've reviewed and met the prerequisites.

1. Create a JSON file that contains your inventory table configuration, and save it (for example, `inventory-config.json`). The following is a sample configuration to enable a new inventory table.

   If you're enabling an inventory table, you can optionally specify an encryption configuration. By default, metadata tables are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3), which you can specify by setting `SseAlgorithm` to `AES256`.

   To encrypt your inventory table with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS), set `SseAlgorithm` to `aws:kms`. You must also set `KmsKeyArn` to the ARN of a customer managed KMS key in the same Region where your general purpose bucket is located.

   ```
   {
     "ConfigurationState": "ENABLED",
   ```

```
  "EncryptionConfiguration": {
    "SseAlgorithm": "aws:kms",
    "KmsKeyArn": "arn:aws:kms:us-east-2:account-id:key/key-id"
  }
}
```

If you want to disable an existing inventory table, use the following configuration:

```
{
  "ConfigurationState": "DISABLED"  }
}
```

2. Use the following command to update the inventory table configuration for your general purpose bucket (for example, *amzn-s3-demo-bucket*):

```
aws s3api update-bucket-metadata-inventory-table-configuration \
--bucket amzn-s3-demo-source-bucket \
--inventory-table-configuration file://./inventory-config.json \
--region us-east-2
```

Using the REST API

You can send REST requests to enable or disable inventory tables. For more information, see UpdateBucketMetadataInventoryTableConfiguration.

Using the AWS SDKs

You can use the AWS SDKs to enable or disable inventory tables in Amazon S3. For information, see the list of supported SDKs.

Enabling or disabling annotation tables

By default, your metadata table configuration contains a *journal table*, which records the events that occur for the objects in your bucket. The journal table is required for each metadata table configuration.

Optionally, you can add an *annotation table* to your metadata table configuration. The annotation table tracks all annotations on objects in your bucket so that you can query annotation data at scale.

The annotation table contains the latest annotations for all objects in your bucket. You can use this table to simplify and speed up business workflows and big data jobs by identifying objects based on their annotations. For example, you can query the annotation table to do the following:

- Find all objects with a specific annotation key or value.
- Create a distribution of annotation keys across objects in your bucket.
- Find all objects that have annotations applied by a specific principal.

If you chose to enable an annotation table for your metadata table configuration, the table goes through a process known as *backfilling*, during which Amazon S3 scans your general purpose bucket to retrieve and populate annotations for all objects that exist in the bucket. Depending on the number of objects in your bucket, this process can take minutes to hours. When the backfilling process is finished, the status of your annotation table changes from **Backfilling** to **Active**. After backfilling is completed, updates to your objects are typically reflected in the annotation table within one hour.

Note

- You're charged for backfilling your annotation table. For more information, see Amazon S3 Pricing.
- You can't pause updates to your annotation table and then resume them. However, you can disable the annotation table configuration. Disabling the annotation table doesn't delete it. The annotation table is retained for your records until you decide to delete it.

  If you've disabled your annotation table and later want to re-enable it, you must first delete the old annotation table from your AWS managed table bucket. When you re-enable the annotation table configuration, Amazon S3 creates a new annotation table, and you're charged again for backfilling the new annotation table.

You can enable or disable annotation tables by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

Prerequisites

If you've disabled your annotation table and now want to re-enable it, you must first manually delete the old annotation table from your AWS managed table bucket. Otherwise, re-enabling the

annotation table fails because an annotation table already exists in the table bucket. To delete your annotation table, see the section called “Delete a metadata table”.

When you re-enable the annotation table configuration, Amazon S3 creates a new annotation table, and you're charged again for backfilling the new annotation table.

Annotation table IAM role

The annotation table requires an IAM role that grants Amazon S3 Metadata permission to read annotations from your bucket. Use the following trust and permissions policies when you create this role.

Trust policy

The following trust policy identifies Amazon S3 Metadata as the service principal that can assume the role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "metadata.s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-bucket"
                }
            }
        }
    ]
}
```

Permissions policy

The following permissions policy grants the IAM role the minimum permissions required to read annotations.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PermissionForGetAnnotation",
      "Effect": "Allow",
      "Action": [
        "s3:GetObjectAnnotation",
        "s3:GetObjectVersionAnnotation"
      ],
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/*"]
    },
    {
      "Sid": "PermissionsForListBucket",
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:ListBucketVersions"
      ],
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"]
    },
    {
      "Sid": "PermissionsForDecryptAnnotation",
      "Effect": "Allow",
      "Action": ["kms:Decrypt"],
      "Resource": ["*"]
    }
  ]
}
```

If your objects use SSE-KMS encryption, the `kms:Decrypt` permission covers decryption for the AWS KMS key used to encrypt the objects. You also need `iam:PassRole` permission to assign the role to the annotation table configuration.

Enable or disable annotation tables

Using the S3 console

To enable or disable annotation tables

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **General purpose buckets**.
3. Choose the general purpose bucket with the metadata table configuration that you want to enable or disable an annotation table for.
4. On the bucket's details page, choose the **Metadata** tab.
5. On the **Metadata** tab, choose **Edit**, then choose **Edit annotation table configuration**.
6. On the **Edit annotation table configuration** page, choose **Enabled** or **Disabled** under **Annotation table**.

> **Note**
>
> Before you choose **Enabled**, make sure that you've reviewed and met the prerequisites.

- If you chose **Enabled**, under **IAM role**, choose one of the following IAM role selection methods:
  - **Create new IAM role** – Amazon S3 creates a new IAM role with the required permissions for the annotation table.
  - **Choose from existing IAM roles** – Select an existing IAM role from the **IAM role** dropdown. The role must have the required permissions to read annotations from your bucket.
  - **Enter IAM role ARN** – Enter the ARN of an existing IAM role that has the required permissions.

  For more information about the required permissions for the annotation table IAM role, see Annotation table IAM role.

  You can also choose whether to encrypt your table with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS). By default, annotation tables are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3).

  If you choose to use SSE-KMS, you must provide a customer managed KMS key in the same Region as your general purpose bucket.

> **Important**
>
> You can set the encryption type for your metadata tables only during table creation. After an AWS managed table is created, you can't change its encryption setting.

- To encrypt your annotation table with SSE-S3 (the default), choose **Don't specify encryption type**.
- To encrypt your annotation table with SSE-KMS, choose **Specify encryption type**. Under **Encryption type**, choose **Server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS)**. Under **AWS KMS key**, either choose from your existing KMS keys, or enter your KMS key ARN. If you don't already have a KMS key, choose **Enter KMS key ARN**, and then choose **Create a KMS key**.

- If you chose **Disabled**, under **After the annotation table is disabled, the table will no longer be updated, and updates can't be resumed**, select the checkbox.

7. Choose **Save changes**.

Using the AWS CLI

To run the following commands, you must have the AWS CLI installed and configured. If you don't have the AWS CLI installed, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

Alternatively, you can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? and Getting started with AWS CloudShell in the *AWS CloudShell User Guide*.

To enable or disable annotation tables by using the AWS CLI

To use the following example commands, replace the *`user input placeholders`* with your own information.

> **Note**
>
> Before enabling an annotation configuration, make sure that you've reviewed and met the prerequisites.

1. Create a JSON file that contains your annotation table configuration, and save it (for example, `annotation-config.json`). The following is a sample configuration to enable a new annotation table.

   If you're enabling an annotation table, you can optionally specify an encryption configuration. By default, metadata tables are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3), which you can specify by setting `SseAlgorithm` to `AES256`.

   To encrypt your annotation table with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS), set `SseAlgorithm` to `aws:kms`. You must also set `KmsKeyArn` to the ARN of a customer managed KMS key in the same Region where your general purpose bucket is located.

   ```
   {
     "ConfigurationState": "ENABLED",
     "Role": "arn:aws:iam::account-id:role/annotation-table-role",
     "EncryptionConfiguration": {
       "SseAlgorithm": "aws:kms",
       "KmsKeyArn": "arn:aws:kms:us-east-2:account-id:key/key-id"
     }
   }
   ```

   The `Role` field is required when enabling an annotation table for the first time or when changing the IAM role. The role must grant Amazon S3 Metadata permission to read annotations from your bucket. For more information, see Annotation table IAM role.

   If you want to disable an existing annotation table, use the following configuration:

   ```
   {
     "ConfigurationState": "DISABLED"  }
   ```

2. Use the following command to update the annotation table configuration for your general purpose bucket (for example, *amzn-s3-demo-bucket*):

```
aws s3api update-bucket-metadata-annotation-table-configuration \
--bucket amzn-s3-demo-bucket \
--annotation-table-configuration file://./annotation-config.json \
--region us-east-2
```

Using the REST API

You can send REST requests to enable or disable annotation tables. For more information, see UpdateBucketMetadataAnnotationTableConfiguration.

Using the AWS SDKs

You can use the AWS SDKs to enable or disable annotation tables in Amazon S3. For information, see the list of supported SDKs.

Viewing metadata table configurations

If you've created a metadata table configuration for a general purpose bucket, you can view information about the configuration, such as whether an inventory table or annotation table has been enabled, or whether journal table record expiration has been enabled. You can also view the status of your journal, inventory, and annotation tables.

You can view your metadata table configuration for a general purpose bucket by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

View a metadata table configuration

Using the S3 console

To view a metadata table configuration

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. Choose the general purpose bucket that contains the metadata table configuration that you want to view.
4. On the bucket's details page, choose the **Metadata** tab.

5. On the **Metadata** tab, scroll down to the **Metadata configuration** section. In the **Journal table**, **Inventory table**, and **Annotation table** sections, you can view various information for these configurations, such as their Amazon Resource Names (ARNs), the status of your tables, and whether you've enabled journal table record expiration, an inventory table, or an annotation table.

Using the AWS CLI

To run the following commands, you must have the AWS CLI installed and configured. If you don't have the AWS CLI installed, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

Alternatively, you can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? and Getting started with AWS CloudShell in the *AWS CloudShell User Guide*.

To view a metadata table configuration by using the AWS CLI

To use the following example command, replace the *`user input placeholders`* with your own information.

1. Use the following command to view the metadata table configuration for your general purpose bucket (for example, *`amzn-s3-demo-bucket`*):

```
aws s3api get-bucket-metadata-configuration \
--bucket amzn-s3-demo-bucket \
--region us-east-2
```

2. View the output of this command to see the status of your metadata table configuration. For example:

```
{
    "GetBucketMetadataConfigurationResult": {
        "MetadataConfigurationResult": {
            "DestinationResult": {
                "TableBucketType": "aws",
                "TableBucketArn": "arn:aws:s3tables:us-east-2:111122223333:bucket/
aws-managed-s3-111122223333-us-east-2",
                "TableNamespace": "b_general-purpose-bucket-name"
```

```
            },
            "JournalTableConfigurationResult": {
                "TableStatus": "ACTIVE",
                "TableName": "journal",
                "TableArn": "arn:aws:s3tables:us-east-2:111122223333:bucket/aws-
managed-s3-111122223333-us-east-2/table/0f01234c-fe7a-492f-a4c7-adec3864ea85",
                "EncryptionConfiguration": {
                    "SseAlgorithm": "AES256"
                },
                "RecordExpiration": {
                    "Expiration": "ENABLED",
                    "Days": 10
                }
            },
            "InventoryTableConfigurationResult": {
                "ConfigurationState": "ENABLED",
                "TableStatus": "BACKFILL_COMPLETE",
                "TableName": "inventory",
                "TableArn": "arn:aws:s3tables:us-east-2:111122223333:bucket/aws-
managed-s3-111122223333-us-east-2/table/e123456-b876-4e5e-af29-bb055922ee4d",
                "EncryptionConfiguration": {
                    "SseAlgorithm": "AES256"
                }
            },
            "AnnotationTableConfigurationResult": {
                "ConfigurationState": "ENABLED",
                "TableStatus": "ACTIVE",
                "TableName": "annotation",
                "TableArn": "arn:aws:s3tables:us-east-2:111122223333:bucket/aws-
managed-s3-111122223333-us-east-2/table/f234567-a890-1b2c-d345-e678901f2345",
                "EncryptionConfiguration": {
                    "SseAlgorithm": "AES256"
                },
                "Role": "arn:aws:iam::111122223333:role/annotation-table-role"
            }
        }
    }
}
```

Using the REST API

You can send REST requests to view a metadata table configuration. For more information, see GetBucketMetadataTableConfiguration.

> **Note**
>
> You can use the V2 `GetBucketMetadataConfiguration` API operation with V1 or V2 metadata table configurations. However, if you try to use the V1 `GetBucketMetadataTableConfiguration` API operation with V2 configurations, you will receive an HTTP `405 Method Not Allowed` error.

Using the AWS SDKs

You can use the AWS SDKs to view a metadata table configuration in Amazon S3. For information, see the list of supported SDKs.

Deleting metadata table configurations

If you want to stop updating the metadata table configuration for an Amazon S3 general purpose bucket, you can delete the metadata table configuration that's attached to your bucket. Deleting a metadata table configuration deletes only the configuration. The AWS managed table bucket and your metadata tables still exist, even if you delete the metadata table configuration. However, the metadata tables will no longer be updated.

> **Note**
>
> If you delete your metadata table configuration and want to re-create a configuration for the same general purpose bucket, you must first manually delete the old journal, inventory, and annotation tables from your AWS managed table bucket. Otherwise, creating the new metadata table configuration fails because those tables already exist. To delete your metadata tables, see the section called “Deleting metadata tables”.

To delete your metadata tables, see the section called “Delete a metadata table”. To delete your table bucket, see Deleting table buckets and DeleteTableBucket in the *Amazon S3 API Reference*.

You can delete a metadata table configuration by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

Delete a metadata table configuration

Using the S3 console

To delete a metadata table configuration

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. Choose the general purpose bucket that you want to remove a metadata table configuration from.
4. On the bucket's details page, choose the **Metadata** tab.
5. On the **Metadata** tab, choose **Delete**.
6. In the **Delete metadata configuration** dialog box, enter **`confirm`** to confirm that you want to delete the configuration. Then choose **Delete**.

Using the AWS CLI

To run the following commands, you must have the AWS CLI installed and configured. If you don't have the AWS CLI installed, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

Alternatively, you can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? and Getting started with AWS CloudShell in the *AWS CloudShell User Guide*.

To delete a metadata table configuration by using the AWS CLI

To use the following example commands, replace the *`user input placeholders`* with your own information.

1. Use the following command to delete the metadata table configuration from your general purpose bucket (for example, *`amzn-s3-demo-bucket`*):

```
aws s3api delete-bucket-metadata-configuration \
--bucket amzn-s3-demo-bucket \
--region us-east-2
```

2. To verify that the configuration was deleted, use the following command:

```
aws s3api get-bucket-metadata-configuration \
--bucket amzn-s3-demo-bucket \
--region us-east-2
```

Using the REST API

You can send REST requests to delete a metadata table configuration. For more information, see DeleteBucketMetadataConfiguration.

> **Note**
>
> You can use the V2 `DeleteBucketMetadataConfiguration` API operation with V1 or V2 metadata table configurations. However, if you try to use the V1 `DeleteBucketMetadataTableConfiguration` API operation with V2 configurations, you will receive an HTTP `405 Method Not Allowed` error.

Using the AWS SDKs

You can use the AWS SDKs to delete a metadata table configuration in Amazon S3. For information, see the list of supported SDKs.

Deleting metadata tables

If you want to delete the metadata tables that you created for an Amazon S3 general purpose bucket, you can delete the metadata tables from your AWS managed table bucket.

> **Important**
>
> - Deleting a table is permanent and can't be undone. Before deleting a table, make sure that you have backed up any important data.
> - Before you delete a metadata table, we recommend that you first delete the associated metadata table configuration on your general purpose bucket. For more information, see the section called “Deleting metadata table configurations”.

To delete your AWS managed table bucket, see Deleting table buckets and DeleteTableBucket in the *Amazon S3 API Reference*. Before you delete your AWS managed table bucket, we recommend that you first delete all metadata table configurations that are associated with this bucket. You must also first delete all metadata tables in the bucket.

You can delete a metadata table by using the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

Delete a metadata table

Using the AWS CLI

To run the following commands, you must have the AWS CLI installed and configured. If you don't have the AWS CLI installed, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

Alternatively, you can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? and Getting started with AWS CloudShell in the *AWS CloudShell User Guide*.

To delete a metadata table configuration by using the AWS CLI

To use the following example commands, replace the *`user input placeholders`* with your own information.

1. Use the following command to delete the metadata table from your AWS managed table bucket:

```
aws s3tables delete-table \
--table-bucket-arn arn:aws:s3tables:us-east-2:111122223333:bucket/aws-s3 \
--namespace b_general-purpose-bucket-name \
--name journal \
--region us-east-2
```

2. To verify that the table was deleted, use the following command:

```
aws s3tables get-table \
--table-bucket-arn arn:aws:s3tables:us-east-2:111122223333:bucket/aws-s3 \
--namespace b_general-purpose-bucket-name \
--name journal \
```

```
--region us-east-2
```

Using the REST API

You can send REST requests to delete a metadata table configuration. For more information, see DeleteTable in the *Amazon S3 API Reference*.

Using the AWS SDKs

You can use the AWS SDKs to delete a metadata table configuration in Amazon S3. For information, see the list of supported SDKs in the *Amazon S3 API Reference*.

Querying metadata tables

Your Amazon S3 Metadata tables are stored in an AWS managed S3 table bucket, which provides storage that's optimized for tabular data. To query your metadata, you can integrate your table bucket with Amazon SageMaker Lakehouse. This integration, which uses the AWS Glue Data Catalog and AWS Lake Formation, allows AWS analytics services to automatically discover and access your table data.

After your table bucket is integrated with the AWS Glue Data Catalog, you can directly query your metadata tables with AWS analytics services such as Amazon Athena, Amazon EMR, and Amazon Redshift. You can also create interactive dashboards with your query data by using AWS Quick.

For more information about integrating your AWS managed S3 table bucket with Amazon SageMaker Lakehouse, see the section called “Integrating S3 Tables with AWS analytics services”.

You can also query your metadata tables with Apache Spark, Apache Trino, and any other application that supports the Apache Iceberg format by using the AWS Glue Iceberg REST endpoint, Amazon S3 Tables Iceberg REST endpoint, or the Amazon S3 Tables Catalog for Apache Iceberg client catalog. For more information about accessing your metadata tables, see the section called “Accessing tables”.

You can analyze your metadata tables with any query engine that supports the Apache Iceberg format. For example, you can query your metadata tables to do the following:

- Discover storage usage patterns and trends
- Audit AWS Key Management Service (AWS KMS) encryption key usage across your objects
- Search for objects by user-defined metadata and object tags

- Understand object metadata changes over time
- Learn when objects are updated or deleted, including the AWS account ID or IP address that made the request

You can also join S3 managed metadata tables and custom metadata tables, allowing you to query across multiple datasets.

Query pricing considerations

Additional pricing applies for running queries on your metadata tables. For more information, see pricing information for the query engine that you're using.

For information on making your queries more cost effective, see Optimizing metadata table query performance.

Topics

- Permissions for querying metadata tables
- Querying metadata tables with AWS analytics services
- Querying metadata tables with open-source query engines
- Optimizing metadata table query performance
- Example metadata table queries

Permissions for querying metadata tables

Before you can query your S3 Metadata journal and live inventory tables, you must have certain S3 Tables permissions. If your metadata tables have been encrypted with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS), you must also have the `kms:Decrypt` permission to decrypt the table data.

When you create your metadata table configuration, your metadata tables are stored in an AWS managed table bucket. All metadata table configurations in your account and in the same Region are stored in a single AWS managed table bucket named `aws-s3`.

To query metadata tables, you can use the following example policy. To use this policy, replace the *`user input placeholders`* with your own information.

```
{
   "Version":"2012-10-17",
```

```
   "Statement":[
      {
         "Sid":"PermissionsToQueryMetadataTables",
         "Effect":"Allow",
         "Action":[
             "s3tables:GetTable",
             "s3tables:GetTableData",
             "s3tables:GetTableMetadataLocation",
             "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3",
            "arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3/table/*",
            "arn:aws:kms:us-east-1:111122223333:key/01234567-89ab-
cdef-0123-456789abcdef"
         ]
       }
    ]
}
```

Querying metadata tables with AWS analytics services

You can query your S3 managed metadata tables with AWS analytics services such as Amazon Athena, Amazon Redshift, and Amazon EMR.

Before you can run queries, you must first integrate the AWS managed S3 table buckets in your AWS account and Region with AWS analytics services.

Querying metadata tables with Amazon Athena

After you integrate your AWS managed S3 table buckets with AWS analytics services, you can start querying your metadata tables in Athena. In your queries, do the following:

- Specify your catalog as `s3tablescatalog/aws-s3` and your database as b_*general_purpose_bucket_name* (which is typically the namespace for your metadata tables).

- Make sure to surround your metadata table namespace names in quotation marks (") or backticks (`), otherwise the query might not work.

For more information, see Querying Amazon S3 tables with Athena.

You can also run queries in Athena from the Amazon S3 console.

Using the S3 console and Amazon Athena

The following procedure uses the Amazon S3 console to access the Athena query editor so that you can query a table with Amazon Athena.

To query a metadata table

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. On the **General purpose buckets** tab, choose the bucket that contains the metadata configuration for the metadata table that you want to query.
4. On the bucket details page, choose the **Metadata** tab.
5. Choose **Query table with Athena**, and then choose one of the sample queries for journal or inventory tables.
6. The Amazon Athena console opens and the Athena query editor appears with a sample query loaded for you. Modify this query as needed for your use case.

   In the query editor, the **Catalog** field should be populated with **s3tablescatalog/aws-s3**. The **Database** field should be populated with the namespace where your table is stored (for example, **b_*general-purpose-bucket-name***).

   > **Note**
   >
   > If you don't see these values in the **Catalog** and **Database** fields, make sure that you've integrated your AWS managed table bucket with AWS analytics services in this Region. For more information, see the section called “Integrating S3 Tables with AWS analytics services”.

7. To run the query, choose **Run**.

   > **Note**
   >
   > - If you receive the error "Insufficient permissions to execute the query. Principal does not have any privilege on specified resource" when you try to run a query in Athena, you must be granted the necessary Lake Formation permissions on the table. For

more information, see the section called “Granting Lake Formation permission on a table or database”.

Also make sure that you have the appropriate AWS Identity and Access Management (IAM) permissions to query metadata tables. For more information, see the section called “Permissions for querying metadata tables”.

- If you receive the error "Iceberg cannot access the requested resource" when you try to run the query, go to the AWS Lake Formation console and make sure that you've granted yourself permissions on the table bucket catalog and database (namespace) that you created. Don't specify a table when granting these permissions. For more information, see the section called “Granting Lake Formation permission on a table or database”.

Querying metadata tables with Amazon Redshift

After you integrate your AWS managed S3 table buckets with AWS analytics services, do the following:

- Create a resource link to your metadata table namespace (typically b_*general_purpose_bucket_name*).
- Make sure to surround your metadata table namespace names in quotation marks (") or backticks (`), otherwise the query might not work.

After that's done, you can start querying your metadata tables in the Amazon Redshift console. For more information, see Accessing Amazon S3 tables with Amazon Redshift.

Querying metadata tables with Amazon EMR

To query your metadata tables by using Amazon EMR, you create an Amazon EMR cluster configured for Apache Iceberg and connect to your metadata tables using Apache Spark. You can set this up by integrating your AWS managed S3 table buckets with AWS analytics services or using the open-source Amazon S3 Tables Catalog for Iceberg client catalog.

Note

When using Apache Spark on Amazon EMR or other third-party engines to query your metadata tables, we recommend that you use the Amazon S3 Tables Iceberg REST

endpoint. Your query might not run successfully if you don't use this endpoint. For more information, see the section called "Accessing tables using the Amazon S3 Tables Iceberg REST endpoint".

For more information, see Accessing Amazon S3 tables with Amazon EMR.

Querying metadata tables with open-source query engines

You can query your S3 managed metadata tables by using open-source query engines, such as Apache Spark. When using Apache Spark on Amazon EMR or other third-party engines to query your metadata tables, we recommend that you use the Amazon S3 Tables Iceberg REST endpoint. Your query might not run successfully if you don't use this endpoint. For more information, see the section called "Accessing tables using the Amazon S3 Tables Iceberg REST endpoint".

Optimizing metadata table query performance

Because S3 Metadata is based on the Apache Iceberg table format, you can optimize the performance and cost of your journal table queries by using specific time ranges.

For example, the following SQL query provides the sensitivity level of new objects in an S3 general purpose bucket:

```
SELECT key, object_tags['SensitivityLevel']
FROM "b_general-purpose-bucket-name"."journal"
WHERE record_type = 'CREATE'
GROUP BY object_tags['SensitivityLevel']
```

This query scans the entire journal table, which might take a long time to run. To improve performance, you can include the `record_timestamp` column to focus on a specific time range. We also recommend using the fully qualified table name, which you can find in the Amazon S3 console on the metadata configuration details page on the general purpose bucket's **Metadata** tab. Here's an updated version of the previous query that looks at new objects from the past month:

```
SELECT key, object_tags['SensitivityLevel']
FROM b_general-purpose-bucket-name"."aws-s3.b_general-purpose-bucket-name.journal"
WHERE record_type = 'CREATE'
AND record_timestamp > (CURRENT_TIMESTAMP - interval '1' month)
GROUP BY object_tags['SensitivityLevel']
```

To improve the performance of queries on inventory tables, make sure that you query only on the minimum columns that you need.

Example metadata table queries

The following examples show how you can get different types information from your S3 Metadata tables by using standard SQL queries.

Remember when using these examples:

- The examples are written to work with Amazon Athena. You might have to modify the examples to work with a different query engine.
- Make sure that you understand how to optimize your queries.
- Replace b_*general-purpose-bucket-name* with the name of your namespace.
- For a full list of supported columns, see the S3 Metadata journal tables schema and the section called "Live inventory tables schema".

Contents

- Journal table example queries
  - Finding objects by file extension
  - Listing object deletions
  - Listing AWS KMS encryption keys used by your objects
  - Listing objects that don't use KMS keys
  - Listing AWS KMS encryption keys used for PUT operations in the last 7 days
  - Listing objects deleted in the last 24 hours by S3 Lifecycle
  - Viewing metadata provided by Amazon Bedrock
  - Understanding the current state of your objects
- Inventory table example queries
  - Discovering datasets that use specific tags
  - Listing objects not encrypted with SSE-KMS
  - Listing objects that aren't encrypted
  - Listing objects generated by Amazon Bedrock
  - Reconciling the inventory table with the journal table
  - Finding the current versions of your objects

- Annotation table example queries
  - Listing all annotations for a specific object
  - Finding all objects that have a specific annotation
  - Joining the annotation table with the inventory table
- Joining custom metadata with S3 metadata tables
- Visualizing metadata table data with AWS Quick

Journal table example queries

You can use the following example queries to query your journal tables.

Finding objects by file extension

The following query returns objects with a specific file extension (`.jpg` in this case):

```
SELECT key FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."journal"
WHERE key LIKE '%.jpg'
AND record_type = 'CREATE'
```

Listing object deletions

The following query returns object deletion events, including the AWS account ID or AWS service principal that made the request:

```
SELECT DISTINCT bucket, key, sequence_number, record_type, record_timestamp, requester,
 source_ip_address, version_id
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."journal"
WHERE record_type = 'DELETE';
```

Listing AWS KMS encryption keys used by your objects

The following query returns the ARNs of the AWS Key Management Service (AWS KMS) keys encrypting your objects:

```
SELECT DISTINCT kms_key_arn
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."journal";
```

Listing objects that don't use KMS keys

The following query returns objects that aren't encrypted with AWS KMS keys:

```
SELECT DISTINCT kms_key_arn
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."journal"
WHERE encryption_status NOT IN ('SSE-KMS', 'DSSE-KMS')
AND record_type = 'CREATE';
```

Listing AWS KMS encryption keys used for PUT operations in the last 7 days

The following query returns the ARNs of the AWS Key Management Service (AWS KMS) keys encrypting your objects:

```
SELECT DISTINCT kms_key_arn
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."journal"
WHERE record_timestamp > (current_date - interval '7' day)
AND kms_key_arn is NOT NULL;
```

Listing objects deleted in the last 24 hours by S3 Lifecycle

The following query returns lists the objects expired in the last day by S3 Lifecycle:

```
SELECT bucket, key, version_id, last_modified_date, record_timestamp, requester
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."journal"
WHERE requester = 's3.amazonaws.com'
AND record_type = 'DELETE'
AND record_timestamp > (current_date - interval '1' day)
```

Viewing metadata provided by Amazon Bedrock

Some AWS services (such as Amazon Bedrock), upload objects to Amazon S3. You can query the object metadata provided by these services. For example, the following query includes the `user_metadata` column to determine if there are objects uploaded by Amazon Bedrock to a general purpose bucket:

```
SELECT DISTINCT bucket, key, sequence_number, record_type, record_timestamp,
 user_metadata
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."journal"
WHERE record_type = 'CREATE'
AND user_metadata['content-source'] = 'AmazonBedrock';
```

If Amazon Bedrock uploaded an object to your bucket, the `user_metadata` column will display the following metadata associated with the object in the query result:

```
user_metadata
{content-additional-params -> requestid="CVK8FWYRW0M9JW65",
 signedContentSHA384="38b060a751ac96384cd9327eb1b1e36a21fdb71114be07434c0cc7bf63f6e1da274edebfe
 content-model-id -> bedrock-model-arn, content-source -> AmazonBedrock}
```

Understanding the current state of your objects

The following query can help you determine the current state of your objects. The query identifies the most recent version of each object, filters out deleted objects, and marks the latest version of each object based on sequence numbers. Results are ordered by the `bucket`, `key`, and `sequence_number` columns.

```
WITH records_of_interest as (
   -- Start with a query that can narrow down the records of interest.
    SELECT * from "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."journal"
),

version_stacks as (
   SELECT *,
          -- Introduce a column called 'next_sequence_number', which is the next larger
          -- sequence_number for the same key version_id in sorted order.
          LEAD(sequence_number, 1) over (partition by (bucket, key,
 coalesce(version_id, '')) order by sequence_number ASC) as next_sequence_number
   from records_of_interest
),

-- Pick the 'tip' of each version stack triple: (bucket, key, version_id).
-- The tip of the version stack is the row of that triple with the largest sequencer.
-- Selecting only the tip filters out any row duplicates.
-- This isn't typical, but some events can be delivered more than once to the table
-- and include rows that might no longer exist in the bucket (since the
-- table contains rows for both extant and extinct objects).
-- In the next subquery, eliminate the rows that contain deleted objects.
current_versions as (
    SELECT * from version_stacks where next_sequence_number is NULL
),

-- Eliminate the rows that are extinct from the bucket by filtering with
-- record_type. An object version has been deleted from the bucket if its tip is
-- record_type==DELETE.
existing_current_versions as (
```

```
    SELECT * from current_versions where not (record_type = 'DELETE' and
 is_delete_marker = FALSE)
),

-- Optionally, to determine which of several object versions is the 'latest',
-- you can compare their sequence numbers. A version_id is the latest if its
-- tip's sequencer is the largest among all other tips in the same key.
with_is_latest as (
    SELECT *,
           -- Determine if the sequence_number of this row is the same as the largest
 sequencer for the key that still exists.
           sequence_number = (MAX(sequence_number) over (partition by (bucket, key)))
 as is_latest_version
    FROM existing_current_versions
)

SELECT * from with_is_latest
ORDER BY bucket, key, sequence_number;
```

Inventory table example queries

You can use the following example queries to query your inventory tables.

Discovering datasets that use specific tags

The following query returns the dataset that uses the specified tags:

```
SELECT *
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."inventory"
WHERE object_tags['key1'] = 'value1'
AND object_tags['key2'] = 'value2';
```

Listing objects not encrypted with SSE-KMS

The following query returns objects that aren't encrypted with SSE-KMS:

```
SELECT key, encryption_status
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."inventory"
WHERE encryption_status != 'SSE-KMS';
```

Listing objects that aren't encrypted

The following query returns objects that aren't encrypted:

```
SELECT bucket, key, version_id
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."inventory"
WHERE encryption_status IS NULL;
```

Listing objects generated by Amazon Bedrock

The following query lists objects that were generated by Amazon Bedrock:

```
SELECT DISTINCT bucket, key, sequence_number, user_metadata
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."inventory"
WHERE user_metadata['content-source'] = 'AmazonBedrock';
```

Reconciling the inventory table with the journal table

The following query generates an inventory-table-like list that's up to date with the current contents of the bucket. More precisely, the resulting list combines the latest snapshot of the inventory table with the latest events in the journal table.

For this query to produce the most accurate results, both the journal and inventory tables must be in Active status.

We recommend using this query for general purpose buckets containing fewer than a billion (10^9) objects.

This example query applies the following simplifications to the list results (compared to the inventory table):

- **Column omissions** – The columns `bucket`, `is_multipart`, `encryption_status`, `is_bucket_key_enabled`, `kms_key_arn`, and `checksum_algorithm` aren't part of the final results. Keeping the set of optional columns to a minimum improves performance.
- **Inclusion of all records** – The query returns all object keys and versions, including the null version (in unversioned or versioning-suspended buckets) and delete markers. For examples of how to filter the results to show only the keys that you're interested in, see the `WHERE` clause at the end of the query.
- **Accelerated reconciliation** – The query could, in rare cases, temporarily report objects that are no longer in the bucket. Those discrepancies are eliminated as soon as the next snapshot of the inventory table becomes available. This behavior is a tradeoff between performance and accuracy.

To run this query in Amazon Athena, make sure to select the `s3tablescatalog/aws-s3` catalog and the b_*general-purpose-bucket-name* database for the general purpose bucket metadata configuration that contains your journal and inventory tables.

```
WITH inventory_time_cte AS (
    SELECT COALESCE(inventory_time_from_property, inventory_time_default) AS
 inventory_time FROM
    (
      SELECT * FROM
        (VALUES (TIMESTAMP '2024-12-01 00:00')) AS T (inventory_time_default)
      LEFT OUTER JOIN
        (
         SELECT from_unixtime(CAST(value AS BIGINT) / 1000.0) AS
 inventory_time_from_property FROM "journal$properties"
         WHERE key = 'aws.s3metadata.oldest-uncoalesced-record-timestamp' LIMIT 1
        )
      ON TRUE
    )
),

working_set AS (
    SELECT
        key,
        sequence_number,
        version_id,
        is_delete_marker,
        size,
        COALESCE(last_modified_date, record_timestamp) AS last_modified_date,
        e_tag,
        storage_class,
        object_tags,
        user_metadata,
        (record_type = 'DELETE' AND NOT COALESCE(is_delete_marker, FALSE)) AS
 _is_perm_delete
    FROM journal j
    CROSS JOIN inventory_time_cte t
    WHERE j.record_timestamp > (t.inventory_time - interval '15' minute)

    UNION ALL

    SELECT
        key,
        sequence_number,
```

```
        version_id,
        is_delete_marker,
        size,
        last_modified_date,
        e_tag,
        storage_class,
        object_tags,
        user_metadata,
        FALSE AS _is_perm_delete
    FROM inventory i
),

updated_inventory AS (
    SELECT * FROM (
        SELECT *,
            MAX(sequence_number) OVER (PARTITION BY key, version_id) AS _supremum_sn
        FROM working_set
    )
    WHERE sequence_number = _supremum_sn
)

SELECT
    key,
    sequence_number,
    version_id,
    is_delete_marker,
    size,
    last_modified_date,
    e_tag,
    storage_class,
    object_tags,
    user_metadata
FROM updated_inventory
-- This filter omits only permanent deletes from the results. Delete markers will still
 be shown.
WHERE NOT _is_perm_delete
-- You can add additional filters here. Examples:
--    AND object_tags['department'] = 'billing'
--    AND starts_with(key, 'reports/')
ORDER BY key ASC, sequence_number DESC;
```

Finding the current versions of your objects

The following query uses the inventory table to generate a new output table that shows which object versions are current. The output table is intentionally similar to an S3 Inventory report. The output table includes an `is_latest` field, which indicates if an object is the current version. The `is_latest` field is equivalent to the **IsLatest** field in an S3 Inventory report.

This query works for general purpose buckets with S3 Versioning in a versioning-enabled or versioning-suspended state.

Prerequisites

The query outputs the results to a new S3 table to support further queries and for higher performance versus outputting rows on screen. Therefore, before running this query, make sure you've met the following conditions. If you choose not to output the results to a new table, you can skip these steps.

- You must have an existing customer-managed table bucket with an existing namespace as a place to output the new table. For more information, see the section called "Create a table bucket" and the section called "Create a namespace".
- To query your new output table, you must set up an access method for querying it. For more information, see the section called "Accessing tables". If you want to query the output table with AWS analytics services such as Amazon Athena, your customer-managed table bucket must be integrated with AWS analytics services. For more information, see the section called "S3 Tables integration overview".

To use this query, replace *`amzn-s3-demo-table-bucket`* with the name of the existing customer-managed table bucket where you want the new output table to be created. Replace *`existing_namespace`* with the name of the namespace where you want the output table to be created in your table bucket. Replace *`new_table`* with the name that you want to use for the output table. Make sure that the name of your output table follows the table naming rules.

To run this query in Amazon Athena, make sure to select the `s3tablescatalog/aws-s3` catalog and the b*`_general-purpose-bucket-name`* database for the general purpose bucket metadata configuration that contains your inventory table.

```
-- If you don't want to output the results to a new table, remove the following two
 lines
-- (everything before the WITH clause).
```

```
CREATE TABLE "s3tablescatalog/amzn-s3-demo-table-
bucket"."existing_namespace"."new_table"
as (
WITH
my_inventory AS (
  SELECT
        bucket,
        key,
        version_id,
        sequence_number,
        is_delete_marker,
        size,
        last_modified_date,
        storage_class
  FROM inventory
-- For prefix filtering, use a WHERE clause with % at the end.
--     WHERE key LIKE 'prefix%'
  ),

inventory_with_is_latest as (
SELECT *,
       ROW_NUMBER() OVER (
         PARTITION BY key
         ORDER BY sequence_number DESC
       ) = 1 AS is_latest
FROM my_inventory
    )

SELECT
        bucket,
        key,
        version_id,
        sequence_number,
        is_delete_marker,
        size,
        last_modified_date,
        storage_class,
        is_latest

FROM inventory_with_is_latest

-- If you want only the current version of each key, uncomment the following WHERE
 clause.
-- WHERE is_latest = TRUE
```

```
-- If you aren't outputting the results to a new table, remove the next line:
);
```

Annotation table example queries

If you have enabled an annotation table in your S3 Metadata configuration, you can query annotation data using Athena or other analytics services that support Apache Iceberg tables. The annotation table uses the naming convention `"s3tablescatalog/aws-s3"."b_`*bucket-name*`"."annotation"`.

Listing all annotations for a specific object

The following query returns all annotations for a specific object:

```
SELECT name, size, last_modified_date, replication_status
FROM "s3tablescatalog/aws-s3"."b_amzn-s3-demo-bucket"."annotation"
WHERE object_key = 'documents/report.pdf'
ORDER BY name
```

Finding all objects that have a specific annotation

The following query returns all objects that have a specific annotation:

```
SELECT object_key, object_version_id, size, last_modified_date
FROM "s3tablescatalog/aws-s3"."b_amzn-s3-demo-bucket"."annotation"
WHERE name = 'classification'
ORDER BY last_modified_date DESC
```

Joining the annotation table with the inventory table

The following query joins the annotation table with the inventory table to combine annotation data with inventory metadata:

```
SELECT i.key, i.storage_class, a.name, a.size, a.text_value
FROM "s3tablescatalog/aws-s3"."b_amzn-s3-demo-bucket"."inventory" i
JOIN "s3tablescatalog/aws-s3"."b_amzn-s3-demo-bucket"."annotation" a
  ON i.bucket = a.bucket
  AND i.key = a.object_key
  AND i.object_join_key = a.object_join_key
WHERE a.name = 'classification'
```

```
LIMIT 10
```

Use the `object_join_key` column when joining annotation and inventory tables to ensure consistency. This column contains a unique identifier for the object, assigned with each new version, or when the null version is created or overwritten. It guarantees that the annotation row corresponds to the correct object version state.

Joining custom metadata with S3 metadata tables

You can analyze data across your AWS managed metadata tables and customer (self-managed) metadata tables. By using a standard SQL `JOIN` operator, you can query data from these multiple sources.

The following example SQL query finds matching records between an AWS managed journal table (`"journal"`) and a self-managed metadata table (*`my_self_managed_metadata_table`*). The query also filters information based on `CREATE` events, which indicate that a new object (or a new version of the object) was written to the bucket. (For more information, see the S3 Metadata journal tables schema.)

```
SELECT *
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."journal" a
JOIN "my_namespace"."my_self_managed_metadata_table" b
ON a.bucket = b.bucket AND a.key = b.key AND a.version_id = b.version_id
WHERE a.record_type = 'CREATE';
```

The following example SQL query finds matching records between an AWS managed inventory table (`"inventory"`) and a self-managed metadata table (*`my_self_managed_metadata_table`*):

```
SELECT *
FROM "s3tablescatalog/aws-s3"."b_general-purpose-bucket-name"."inventory" a
JOIN "my_namespace"."my_self_managed_metadata_table" b
ON a.bucket = b.bucket AND a.key = b.key AND a.version_id = b.version_id;
```

Visualizing metadata table data with AWS Quick

With AWS Quick, you can create interactive dashboards to analyze and visualize SQL query results about your S3 managed metadata tables. Quick dashboards can help you monitor statistics, track changes, and get operational insights about your metadata tables.

A dashboard about your journal table might show you:

- What's the percentage of object uploads compared to deletions?
- Which objects were deleted by S3 Lifecycle in the past 24 hours?
- Which IP addresses did the most recent PUT requests come from?

A dashboard about your inventory table might show you:

- How many objects are in different storage classes?
- What percentage of your storage data is small objects compared to large objects?
- What types of objects are in my bucket?

After you integrate your S3 table buckets with AWS analytics services, you can create datasets from your metadata tables and work with them in Amazon Quick using SPICE or direct SQL queries from your query engine. Quick supports Amazon Athena and Amazon Redshift as data sources.

For more information, see Visualizing table data with AWS Quick.

Troubleshooting S3 Metadata

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon S3 Metadata.

I'm unable to delete my AWS managed table bucket and metadata tables

Before you can delete a metadata table, you must first delete the associated metadata table configuration on your general purpose bucket. For more information, see the section called "Deleting metadata table configurations".

Before you can delete your AWS managed table bucket, you must delete all metadata table configurations that are associated with this bucket and all metadata tables in the bucket. For more information, see the section called "Deleting metadata table configurations" and the section called "Deleting metadata tables".

I'm unable to set or change the encryption settings for my AWS managed metadata table

When you create your metadata table configuration, you can choose to encrypt your AWS managed metadata tables with server-side encryption using AWS Key Management Service (AWS KMS) keys

(SSE-KMS). If you choose to use SSE-KMS, you must provide a customer managed KMS key in the same Region as your general purpose bucket. You can set the encryption type for your tables only during table creation. After an AWS managed table is created, you can't change its encryption setting. To specify SSE-KMS for your metadata tables, you must have certain permissions. For more information, see Permissions for SSE-KMS.

The encryption setting for a metadata table takes precedence over the default bucket-level encryption setting. If you don't specify encryption for a table, it will inherit the default encryption setting from the bucket.

By default, AWS managed table buckets are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3). After you create your first metadata configuration, you can set the default encryption setting for the AWS managed table bucket to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS). For more information, see Encryption for AWS managed table buckets and the section called “Specifying SSE-KMS”.

When I try to re-create my metadata table configuration, I get an error

Deleting a metadata table configuration deletes only the configuration. The AWS managed table bucket and your metadata tables still exist, even if you delete the metadata table configuration.

If you delete your metadata table configuration and want to re-create a configuration for the same general purpose bucket, you must first manually delete the old journal, inventory, and annotation tables from your AWS managed table bucket. Otherwise, creating the new metadata table configuration fails because those tables already exist.

To delete your metadata tables, see the section called “Deleting metadata tables”.

I can't enable an inventory table on my configuration

If you created your S3 Metadata configuration before July 15, 2025, you can't enable an inventory table on that configuration. We recommend that you delete and re-create your configuration so that you can create an inventory table and expire journal table records. For more information, see the section called “Enabling inventory and annotation tables on metadata configurations created before July 15, 2025”.

I can't enable an annotation table on my configuration

If you created your S3 Metadata configuration before July 15, 2025, you can't enable an annotation table on that configuration. We recommend that you delete and re-create your configuration

so that you can create an annotation table, create an inventory table, and expire journal table records. For more information, see the section called “Enabling inventory and annotation tables on metadata configurations created before July 15, 2025”.

I can't enable journal table record expiration on my configuration

If you created your S3 Metadata configuration before July 15, 2025, you can't enable journal table record expiration on that configuration. We recommend that you delete and re-create your configuration so that you can expire journal table records and create an inventory table. For more information, see the section called “Enabling inventory and annotation tables on metadata configurations created before July 15, 2025”.

I can't query my metadata tables

If you're unable to query your metadata tables, check the following:

- When you're using Amazon Athena or Amazon Redshift to query your metadata tables, you must surround your metadata table namespace names in quotation marks (") or backticks (`), otherwise the query might not work.
- When using Apache Spark on Amazon EMR or other third-party engines to query your metadata tables, we recommend that you use the Amazon S3 Tables Iceberg REST endpoint. Your query might not run successfully if you don't use this endpoint. For more information, see the section called “Accessing tables using the Amazon S3 Tables Iceberg REST endpoint”.
- Make sure that you have the appropriate AWS Identity and Access Management (IAM) permissions to query metadata tables. For more information, see the section called “Permissions for querying metadata tables”.
- If you're using Amazon Athena and receive errors when you try to run your queries, do the following:
  - If you receive the error "Insufficient permissions to execute the query. Principal does not have any privilege on specified resource" when you try to run a query in Athena, you must be granted the necessary Lake Formation permissions on the table. For more information, see the section called “Granting Lake Formation permission on a table or database”.
  - If you receive the error "Iceberg cannot access the requested resource" when you try to run the query, go to the AWS Lake Formation console and make sure that you've granted yourself permissions on the table bucket catalog and database (namespace) that you created. Don't specify a table when granting these permissions. For more information, see the section called “Granting Lake Formation permission on a table or database”.

I'm receiving 405 errors when I try to use certain S3 Metadata AWS CLI commands and API operations

Calling the V1 `GetBucketMetadataTableConfiguration` API operation or using the `get-bucket-metadata-table-configuration` AWS Command Line Interface (AWS CLI) command against a V2 metadata table configuration results in an HTTP `405 Method Not Allowed` error. Likewise, calling the V1 `DeleteBucketMetadataTableConfiguration` API operation or using the `delete-bucket-metadata-table-configuration` AWS CLI command also causes a 405 error.

You can use the V2 `GetBucketMetadataConfiguration` API operation or the `get-bucket-metadata-configuration` AWS CLI command against a V1 or V2 metadata table configuration. Likewise, you can use the V2 `DeleteBucketMetadataConfiguration` API operation or the `delete-bucket-metadata-configuration` AWS CLI command against a V1 or V2 metadata table configuration.

We recommend updating your processes to use the new V2 API operations (`CreateBucketMetadataConfiguration`, `GetBucketMetadataConfiguraion`, and `DeleteBucketMetadataConfiguration`) instead of the V1 API operations. For more information about migrating from V1 of S3 Metadata to V2, see the section called “Enabling inventory and annotation tables on metadata configurations created before July 15, 2025”.

To determine whether your configuration is V1 or V2, you can look at the following attribute of your `GetBucketMetadataConfiguration` API response. An AWS managed bucket type (`"aws"`) indicates a V2 configuration, and a customer-managed bucket type (`"customer"`) indicates a V1 configuration.

```
"MetadataTableConfigurationResult": {
            "TableBucketType": ["aws" | "customer"]
```

For more information, see the section called “Viewing metadata table configurations”.

Uploading objects

When you upload a file to Amazon S3, it is stored as an S3 *object*. Objects consist of the file data and metadata that describes the object. You can have an unlimited number of objects in a bucket. Before you can upload files to an Amazon S3 bucket, you need write permissions for the bucket. For more information about access permissions, see Identity and Access Management for Amazon S3.

You can upload any file type—images, backups, data, movies, and so on—into an S3 bucket. The maximum size of a file that you can upload by using the Amazon S3 console is 160 GB. To upload a file larger than 160 GB, use the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

If you upload an object with a key name that already exists in a versioning-enabled bucket, Amazon S3 creates another version of the object instead of replacing the existing object. For more information about enabling versioning, see Enabling versioning on buckets.

Depending on the size of the data that you're uploading, Amazon S3 offers the following options:

- **Upload an object in a single operation by using the AWS SDKs, REST API, or AWS CLI** – With a single PUT operation, you can upload a single object up to 5 GB in size.
- **Upload a single object by using the Amazon S3 console –** With the Amazon S3 console, you can upload a single object up to 160 GB in size.
- **Upload an object in parts by using the AWS SDKs, REST API, or AWS CLI –** Using the multipart upload API operation, you can upload a single large object, up to 50 TB in size.

  The multipart upload API operation is designed to improve the upload experience for larger objects. You can upload an object in parts. These object parts can be uploaded independently, in any order, and in parallel. You can use a multipart upload for objects from 5 MB to 50 TB in size. For more information, see Uploading and copying objects using multipart upload in Amazon S3.

To upload files greater than 5 TB, use the S3 Transfer Manager in the Java v1/v2, Python, or AWS CLI SDKs. For the best performance, use the latest AWS Common Runtime (CRT) with these SDKs, which has been optimized for better resource utilization.

When uploading large objects from memory stream, CRT buffers each part up to 5 GB in memory, limiting overall throughput by allocated memory. You can adjust the CRT memory limit using configuration options such as `maxNativeMemoryLimitInBytes` for Java SDK. For uploads from disk, CRT automatically switches to direct disk streaming instead of intermediate part buffering, improving memory usage. This behavior is automatically enabled for large objects, but can also be enabled for smaller files via request parameters such as `should_stream` for AWS CLI and `CRT_MEMORY_BUFFER_DISABLED` for Java SDK.

When you upload an object, the object is automatically encrypted using server-side encryption with Amazon S3 managed keys (SSE-S3) by default. When you download it, the object is decrypted.

For more information, see Setting default server-side encryption behavior for Amazon S3 buckets and Protecting data with encryption.

When you're uploading an object, if you want to use a different type of default encryption, you can also specify server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) in your S3 PUT requests or set the default encryption configuration in the destination bucket to use SSE-KMS to encrypt your data. For more information about SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS). If you want to use a KMS key that is owned by a different account, you must have permission to use the key. For more information about cross-account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*.

If you encounter an Access Denied (403 Forbidden) error in Amazon S3, see Troubleshoot access denied (403 Forbidden) errors in Amazon S3 to learn more about its common causes.

Upload an object

Using the S3 console

This procedure explains how to upload objects and folders to an Amazon S3 bucket by using the console.

When you upload an object, the object key name is the file name and any optional prefixes. In the Amazon S3 console, you can create folders to organize your objects. In Amazon S3, folders are represented as prefixes that appear in the object key name. If you upload an individual object to a folder in the Amazon S3 console, the folder name is included in the object key name.

For example, if you upload an object named `sample1.jpg` to a folder named `backup`, the key name is `backup/sample1.jpg`. However, the object is displayed in the console as `sample1.jpg` in the `backup` folder. For more information about key names, see Working with object metadata.

> **Note**
>
> If you rename an object or change any of the properties in the Amazon S3 console, for example **Storage Class**, **Encryption**, or **Metadata**, a new object is created to replace the old one. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. The role that changes the property also becomes the owner of the new object (or object version).

When you upload a folder, Amazon S3 uploads all of the files and subfolders from the specified folder to your bucket. It then assigns an object key name that is a combination of the uploaded file name and the folder name. For example, if you upload a folder named `/images` that contains two files, `sample1.jpg` and `sample2.jpg`, Amazon S3 uploads the files and then assigns the corresponding key names, `images/sample1.jpg` and `images/sample2.jpg`. The key names include the folder name as a prefix. The Amazon S3 console displays only the part of the key name that follows the last `/`. For example, within an `images` folder, the `images/sample1.jpg` and `images/sample2.jpg` objects are displayed as `sample1.jpg` and a `sample2.jpg`.

To upload folders and files to an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want to upload your folders or files to.
4. Choose **Upload**.
5. In the **Upload** window, do one of the following:
   - Drag and drop files and folders to the **Upload** window.
   - Choose **Add file** or **Add folder**, choose the files or folders to upload, and choose **Open**.
6. To enable versioning, under **Destination**, choose **Enable Bucket Versioning**.
7. To upload the listed files and folders without configuring additional upload options, choose **Upload**.

   Amazon S3 uploads your objects and folders. When the upload is finished, you see a success message on the **Upload: status** page.

To configure additional object properties

1. To change access control list permissions, choose **Permissions**.
2. Under **Access control list (ACL)**, edit the permissions.

   For information about object access permissions, see Using the S3 console to set ACL permissions for an object. You can grant read access to your objects to the public (everyone in the world) for all of the files that you're uploading. However, we recommend not changing

the default setting for public read access. Granting public read access is applicable to a small subset of use cases, such as when buckets are used for websites. You can always change the object permissions after you upload the object.

3. To configure other additional properties, choose **Properties**.
4. Under **Storage class**, choose the storage class for the files that you're uploading.

   For more information about storage classes, see Understanding and managing Amazon S3 storage classes.
5. To update the encryption settings for your objects, under **Server-side encryption settings**, do the following.

   a. Choose **Specify an encryption key**.

   b. Under **Encryption settings**, choose **Use bucket settings for default encryption** or **Override bucket settings for default encryption**.

   c. If you chose **Override bucket settings for default encryption**, you must configure the following encryption settings.

   - To encrypt the uploaded files by using keys that are managed by Amazon S3, choose **Amazon S3 managed key (SSE-S3)**.

     For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).
   - To encrypt the uploaded files by using keys stored in AWS Key Management Service (AWS KMS), choose **AWS Key Management Service key (SSE-KMS)**. Then choose one of the following options for **AWS KMS key**:
     - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose your **KMS key** from the list of available keys.

       Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
     - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and then enter your KMS key ARN in the field that appears.
     - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that is not listed, you must enter your KMS key ARN. If you want to use a KMS key that is owned by a different account, you must first have permission to use the key and then you must enter the KMS key ARN.
>
> Amazon S3 supports only symmetric encryption KMS keys, and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

6. To use additional checksums, choose **On**. Then for **Checksum function**, choose the function that you would like to use. Amazon S3 calculates and stores the checksum value after it receives the entire object. You can use the **Precalculated value** box to supply a precalculated value. If you do, Amazon S3 compares the value that you provided to the value that it calculates. If the two values do not match, Amazon S3 generates an error.

   Additional checksums enable you to specify the checksum algorithm that you would like to use to verify your data. For more information about additional checksums, see Checking object integrity in Amazon S3.

7. To add tags to all of the objects that you are uploading, choose **Add tag**. Enter a tag name in the **Key** field. Enter a value for the tag.

   Object tagging gives you a way to categorize storage. Each tag is a key-value pair. Key and tag values are case sensitive. You can have up to 10 tags per object. A tag key can be up to 128 Unicode characters in length, and tag values can be up to 255 Unicode characters in length. For more information about object tags, see Tagging your objects.

8. To add metadata, choose **Add metadata**.

   a. Under **Type**, choose **System defined** or **User defined**.

      For system-defined metadata, you can select common HTTP headers, such as **Content-Type** and **Content-Disposition**. For a list of system-defined metadata and information

about whether you can add the value, see System-defined object metadata. Any metadata starting with the prefix `x-amz-meta-` is treated as user-defined metadata. User-defined metadata is stored with the object and is returned when you download the object. Both the keys and their values must conform to US-ASCII standards. User-defined metadata can be as large as 2 KB. For more information about system-defined and user-defined metadata, see Working with object metadata.

b. For **Key**, choose a key.

c. Type a value for the key.

9. To upload your objects, choose **Upload**.

   Amazon S3 uploads your object. When the upload completes, you can see a success message on the **Upload: status** page.

10. Choose **Exit**.

Using the AWS CLI

You can send a PUT request to upload an object of up to 5 GB in a single operation. For more information, see the `PutObject` example in the *AWS CLI Command Reference*.

Using the REST API

You can send REST requests to upload an object. You can send a PUT request to upload data in a single operation. For more information, see PUT Object.

Using the AWS SDKs

For examples of how to upload an object with the AWS SDKs, see Code Examples in the *Amazon Simple Storage Service API Reference*.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon Simple Storage Service API Reference*.

Prevent uploading objects with identical key names

You can check for the existence of an object in your bucket before creating it using a conditional write on upload operations. This can prevent overwrites of existing data. Conditional writes will validate there is no existing object with the same key name already in your bucket while uploading.

You can use conditional writes for PutObject or CompleteMultipartUpload requests.

For more information about conditional requests see, Add preconditions to S3 operations with conditional requests.

Uploading and copying objects using multipart upload in Amazon S3

Multipart upload allows you to upload a single object to Amazon S3 as a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently, and in any order. For uploads, your updated AWS client automatically calculates a checksum of the object and sends it to Amazon S3 along with the size of the object as a part of the request. If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles them to create the object. It's a best practice to use multipart upload for objects that are 100 MB or larger instead of uploading them in a single operation.

Using multipart upload provides the following advantages:

- **Improved throughput** – You can upload parts in parallel to improve throughput.
- **Quick recovery from any network issues** – Smaller part size minimizes the impact of restarting a failed upload due to a network error.
- **Pause and resume object uploads** – You can upload object parts over time. After you initiate a multipart upload, there is no expiry; you must explicitly complete or stop the multipart upload.
- **Begin an upload before you know the final object size** – You can upload an object as you create it.

We recommend that you use multipart upload in the following ways:

- If you upload large objects over a stable high-bandwidth network, use multipart upload to maximize the use of your available bandwidth by uploading object parts in parallel for multi-threaded performance.
- If you upload over a spotty network, use multipart upload to increase resiliency against network errors by avoiding upload restarts. When using multipart upload, you only need to retry uploading the parts that are interrupted during the upload. You don't need to restart uploading your object from the beginning.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets. For more information about using multipart upload with S3 Express One Zone and directory buckets, see Using multipart uploads with directory buckets.

Multipart upload process

Multipart upload is a three-step process: You initiate the upload, upload the object parts, and—after you've uploaded all the parts—complete the multipart upload. Upon receiving the complete multipart upload request, Amazon S3 constructs the object from the uploaded parts, and you can access the object just as you would any other object in your bucket.

You can list all of your in-progress multipart uploads or get a list of the parts that you have uploaded for a specific multipart upload. Each of these operations is explained in this section.

Multipart upload initiation

When you send a request to initiate a multipart upload, make sure to specify a checksum type. Amazon S3 will then return a response with an upload ID, which is a unique identifier for your multipart upload. This upload ID is required when you upload parts, list parts, complete an upload, or stop an upload. If you want to provide metadata describing the object being uploaded, you must provide it in the request to initiate the multipart upload. Anonymous users cannot initiate multipart uploads.

Parts upload

When uploading a part, you must specify a part number in addition to the upload ID. You can choose any part number between 1 and 10,000. A part number uniquely identifies a part and its position in the object you are uploading. The part number that you choose doesn't need to be in a consecutive sequence (for example, it can be 1, 5, and 14). Be aware that if you upload a new part using the same part number as a previously uploaded part, the previously uploaded part gets overwritten.

When you upload a part, Amazon S3 returns the checksum algorithm type with the checksum value for each part as a header in the response. For each part upload, you must record the part number and the ETag value. You must include these values in the subsequent request to complete

the multipart upload. Each part will have its own ETag at the time of upload. However, once the multipart upload is complete and all parts are consolidated, all parts belong to one ETag as a checksum of checksums.

> **Important**
>
> After you initiate a multipart upload and upload one or more parts, you must either complete or stop the multipart upload to stop incurring charges for storage of the uploaded parts. Only *after* you complete or stop a multipart upload will Amazon S3 free up the parts storage and stop billing you for the parts storage.
> After stopping a multipart upload, you can't upload any part using that upload ID again. If part uploads were in progress, they can still succeed or fail even after you stop the upload. To make sure you free all storage consumed by all parts, you must stop a multipart upload only after all part uploads have completed.

Multipart upload completion

When you complete a multipart upload, Amazon S3 creates an object by concatenating the parts in ascending order based on the part number. If any object metadata was provided in the *initiate multipart upload* request, Amazon S3 associates that metadata with the object. After a successful *complete* request, the parts no longer exist.

Your *complete multipart upload* request must include the upload ID and a list of part numbers and their corresponding ETag values. The Amazon S3 response includes an ETag that uniquely identifies the combined object data. This ETag is not necessarily an MD5 hash of the object data.

When you provide a full object checksum during a multipart upload, the AWS SDK passes the checksum to Amazon S3, and S3 validates the object integrity server-side, comparing it to the received value. Then, S3 stores the object if the values match. If the two values don't match, Amazon S3 fails the request with a `BadDigest` error. The checksum of your object is also stored in object metadata that you'll later use to validate an object's data integrity.

Sample multipart upload calls

For this example, assume that you're generating a multipart upload for a 100 GB file. In this case, you would have the following API calls for the entire process. There would be a total of 1,002 API calls.

- A `CreateMultipartUpload` call to start the process.

- 1,000 individual `UploadPart` calls, each uploading a part of 100 MB, for a total size of 100 GB.
- A `CompleteMultipartUpload` call to complete the process.

Multipart upload listings

You can list the parts of a specific multipart upload or all in-progress multipart uploads. The list parts operation returns the parts information that you uploaded for a specific multipart upload. For each list parts request, Amazon S3 returns the parts information for the specified multipart upload, up to a maximum of 1,000 parts. If there are more than 1,000 parts in the multipart upload, you must send a series of list part requests to retrieve all of the parts. Note that the returned list of parts doesn't include parts that haven't finished uploading. Using the *list multipart uploads* operation, you can obtain a list of multipart uploads that are in progress.

An in-progress multipart upload is an upload that you have initiated, but have not yet completed or stopped. Each request returns at most 1,000 multipart uploads. If there are more than 1,000 multipart uploads in progress, you must send additional requests to retrieve the remaining multipart uploads. Use the returned listing only for verification.

> **Important**
>
> Do not use the result of this listing when sending a *complete multipart upload* request. Instead, maintain your own list of the part numbers that you specified when uploading parts and the corresponding ETag values that Amazon S3 returns.

Checksums with multipart upload operations

When you upload an object to Amazon S3, you can specify a checksum algorithm for Amazon S3 to use. By default, the AWS SDK and S3 console use an algorithm for all object uploads, which you can override. If you're using an older SDK and your uploaded object doesn't have a specified checksum, Amazon S3 automatically uses the CRC-64/NVME (`CRC64NVME`) checksum algorithm. (This is also the recommended option for efficient data integrity verification.) When using CRC-64/NVME, Amazon S3 calculates the checksum of the full object after the multipart or single part upload is complete. The CRC-64/NVME checksum algorithm is used to calculate either a direct checksum of the entire object, or a checksum of the checksums, for each individual part.

After you upload an object to S3 using multipart upload, Amazon S3 calculates the checksum value for each part, or for the full object—and stores the values. You can use the S3 API or AWS SDK to retrieve the checksum value in the following ways:

- For individual parts, you can use `GetObject` or `HeadObject`. If you want to retrieve the checksum values for individual parts of multipart uploads while they're still in process, you can use `ListParts`.
- For the entire object, you can use `PutObject`. If you want to perform a multipart upload with a full object checksum, use `CreateMultipartUpload` and `CompleteMultipartUpload` by specifying the full object checksum type. To validate the checksum value of the entire object or to confirm which checksum type is being used in the multipart upload, use `ListParts`.

> **Important**
>
> If you're using a multipart upload with **Checksums**, the part numbers for each part upload (in the multipart upload) must use consecutive part numbers and begin with 1. When using **Checksums**, if you try to complete a multipart upload request with nonconsecutive part numbers, Amazon S3 generates an `HTTP 500 Internal Server` error.

For more information about how checksums work with multipart upload objects, see Checking object integrity in Amazon S3.

For an end-to-end procedure that demonstrates how to upload an object using multipart upload with an additional checksum, see Tutorial: Upload an object through multipart upload and verify its data integrity.

Concurrent multipart upload operations

In a distributed development environment, it is possible for your application to initiate several updates on the same object at the same time. Your application might initiate several multipart uploads using the same object key. For each of these uploads, your application can then upload parts and send a complete upload request to Amazon S3 to create the object. When the buckets have S3 Versioning enabled, completing a multipart upload always creates a new version. When you initiate multiple multipart uploads that use the same object key in a versioning-enabled bucket, the current version of the object is determined by which upload started most recently (`createdDate`).

For example, you start a `CreateMultipartUpload` request for an object at 10:00 AM. Then, you submit a second `CreateMultipartUpload` request for the same object at 11:00 AM. Because the second request was submitted the most recently, the object uploaded by the 11:00 AM request becomes the current version, even if the first upload is completed after the second one. For buckets that don't have versioning enabled, it's possible that any other request received between the time when the multipart upload is initiated and when it completes, the other request might take precedence.

Another example of when a concurrent multipart upload request can take precedence is if another operation deletes a key after you initiate a multipart upload with that key. Before you complete the operation, the complete multipart upload response might indicate a successful object creation without you ever seeing the object.

Prevent uploading objects with identical key names during multipart upload

You can check for the existence of an object in your bucket before creating it using a conditional write on upload operations. This can prevent overwrites of existing data. Conditional writes will validate that there is no existing object with the same key name already in your bucket while uploading.

You can use conditional writes for PutObject or CompleteMultipartUpload requests.

For more information, see How to prevent object overwrites with conditional writes.

Multipart upload and pricing

After you initiate a multipart upload, Amazon S3 retains all the parts until you either complete or stop the upload. Throughout its lifetime, you are billed for all storage, bandwidth, and requests for this multipart upload and its associated parts.

These parts are billed according to the storage class specified when the parts are uploaded. However, you will not be billed for these parts if they're uploaded to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive. In-progress multipart parts for a PUT request to the S3 Glacier Flexible Retrieval storage class are billed as S3 Glacier Flexible Retrieval staging storage at S3 Standard storage rates until the upload completes. In addition, both `CreateMultipartUpload` and `UploadPart` are billed at S3 Standard rates. Only the `CompleteMultipartUpload` request is billed at the S3 Glacier Flexible Retrieval rate. Similarly, in-progress multipart parts for a PUT to the S3 Glacier Deep Archive storage class are billed as S3 Glacier Flexible Retrieval staging storage at S3 Standard storage rates until the upload completes, with only the `CompleteMultipartUpload` request charged at S3 Glacier Deep Archive rates.

If you stop the multipart upload, Amazon S3 deletes upload artifacts and all parts that you uploaded. You will not be billed for those artifacts. There are no early delete charges for deleting incomplete multipart uploads regardless of storage class specified. For more information about pricing, see Amazon S3 pricing.

> **Note**
>
> To minimize your storage costs, we recommend that you configure a lifecycle rule to delete incomplete multipart uploads after a specified number of days by using the `AbortIncompleteMultipartUpload` action. For more information about creating a lifecycle rule to delete incomplete multipart uploads, see Configuring a bucket lifecycle configuration to delete incomplete multipart uploads.

API support for multipart upload

The following sections in the *Amazon Simple Storage Service API Reference* describe the REST API for multipart upload.

For a multipart upload walkthrough that uses AWS Lambda functions, see Uploading large objects to Amazon S3 using multipart upload and transfer acceleration.

- Create Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- Abort Multipart Upload
- List Parts
- List Multipart Uploads

AWS Command Line Interface support for multipart upload

The following topics in the AWS Command Line Interface describe the operations for multipart upload.

- Initiate Multipart Upload
- Upload Part

- Upload Part (Copy)
- Complete Multipart Upload
- Abort Multipart Upload
- List Parts
- List Multipart Uploads

AWS SDK support for multipart upload

You can use an AWS SDKs to upload an object in parts. For a list of AWS SDKs supported by API action see:

- Create Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- Abort Multipart Upload
- List Parts
- List Multipart Uploads

Multipart upload API and permissions

You must have the necessary permissions to use the multipart upload operations. You can use access control lists (ACLs), the bucket policy, or the user policy to grant individuals permissions to perform these operations. The following table lists the required permissions for various multipart upload operations when using ACLs, a bucket policy, or a user policy.

Action	Required permissions
Create Multipart Upload	You must be allowed to perform the `s3:PutObject` action on an object to create a multipart upload request. The bucket owner can allow other principals to perform the `s3:PutObject` action.

Action	Required permissions
Initiate Multipart Upload	You must be allowed to perform the `s3:PutObject` action on an object to initiate a multipart upload. The bucket owner can allow other principals to perform the `s3:PutObject` action.
Initiator	Container element that identifies who initiated the multipart upload. If the initiator is an AWS account, this element provides the same information as the Owner element. If the initiator is an IAM user, this element provides the user ARN and display name.
Upload Part	You must be allowed to perform the `s3:PutObject` action on an object to upload a part. The bucket owner must allow the initiator to perform the `s3:PutObject` action on an object in order for the initiator to upload a part for that object.
Upload Part (Copy)	You must be allowed to perform the `s3:PutObject` action on an object to upload a part. Because you are uploading a part from an existing object, you must be allowed `s3:GetObject` on the source object. For the initiator to upload a part for an object, the owner of the bucket must allow the initiator to perform the `s3:PutObject` action on the object.
Complete Multipart Upload	You must be allowed to perform the `s3:PutObject` action on an object to complete a multipart upload. The bucket owner must allow the initiator to perform the `s3:PutObject` action on an object in order for the initiator to complete a multipart upload for that object.

Action	Required permissions
Stop Multipart Upload	You must be allowed to perform the `s3:AbortMultipartUpload` action to stop a multipart upload. By default, the bucket owner and the initiator of the multipart upload are allowed to perform this action as a part of IAM and S3 bucket polices. If the initiator is an IAM user, that user's AWS account is also allowed to stop that multipart upload. With VPC endpoint policies, the initiator of the multipart upload doesn't automatically gain the permission to perform the `s3:AbortMultipartUpload` action. In addition to these defaults, the bucket owner can allow other principals to perform the `s3:AbortMultipartUpload` action on an object. The bucket owner can deny any principal the ability to perform the `s3:AbortMultipartUpload` action.
List Parts	You must be allowed to perform the `s3:ListMultipartUploadParts` action to list parts in a multipart upload. By default, the bucket owner has permission to list parts for any multipart upload to the bucket. The initiator of the multipart upload has the permission to list parts of the specific multipart upload. If the multipart upload initiator is an IAM user, the AWS account controlling that IAM user also has permission to list parts of that upload. In addition to these defaults, the bucket owner can allow other principals to perform the `s3:ListMultipartUploadParts` action on an object. The bucket owner can also deny any principal the ability to perform the `s3:ListMultipartUploadParts` action.
List Multipart Uploads	You must be allowed to perform the `s3:ListBucketMultipartUploads` action on a bucket to list multipart uploads in progress to that bucket. In addition to the default, the bucket owner can allow other principals to perform the `s3:ListBucketMultipartUploads` action on the bucket.

Action	Required permissions
AWS KMS Encrypt and Decrypt related permissions	To perform a multipart upload with encryption using an AWS Key Management Service (AWS KMS) KMS key, the requester must have the following permissions: • The `kms:Decrypt` and `kms:GenerateDataKey` actions on the key. • The `kms:GenerateDataKey` action for the CreateMultipartUpload API. • The `kms:Decrypt` action on the UploadPart and UploadPartCopy APIs. These permissions are required because Amazon S3 must decrypt and read data from the encrypted file parts before it completes the multipart upload. The `kms:Decrypt` permission, and the server-side encryption with customer-provided encryption keys, are also required for you to obtain an object's checksum value. If you don't have these required permissions when you use the CompleteMultipartUpload API, the object is created without a checksum value. If your IAM user or role is in the same AWS account as the KMS key, then validate that you have permissions on both the key and IAM policies. If your IAM user or role belongs to a different account than the KMS key, then you must have the permissions on both the key policy and your IAM user or role.
SSE-C (server-side encryption with customer-provided encryption keys)	When you use the CompleteMultipartUpload API, you must provide the SSE-C (server-side encryption with customer-provided encryption keys), or your object will be created without a checksum, and no checksum value is returned.

For information on the relationship between ACL permissions and permissions in access policies, see Mapping of ACL permissions and access policy permissions. For information about IAM users, roles, and best practices, see IAM identities (users, user groups, and roles) in the *IAM User Guide*.

Checksums with multipart upload operations

There are three Amazon S3 APIs that are used to perform the actual multipart upload: `CreateMultipartUpload`, `UploadPart`, and `CompleteMultipartUpload`. The following table indicates which checksum headers and values must be provided for each of the APIs:

Checksum algorithm	Checksum type	CreateMultipartUpload	UploadPart	CompleteMultipartUpload

CRC-64/NVME (CRC64NVME)	Full object	Required headers:
		`x-amz-checksum-algorithm`
			Optional headers:
			`x-amz-checksum-crc64nvme`
				Optional headers:
				`x-amz-checksum-algorithm`
				`x-amz-crc64`

CRC-32 (CRC32)
CRC 32-C (CRC32C)
	Full object	Required headers:
		`x-amz-checksum-algorithm`
		`x-amz-checksum-type`
			Optional headers:
			`x-amz-checksum-crc64nvme`
				Optional headers:
				`x-amz-checksum-algorithm`
				`x-amz-crc32`
				`x-amz-crc32c`

CRC-32 (CRC32) CRC-32C (CRC32C) SHA-1 (SHA1) SHA-256 (SHA256)	Composite	Required headers: `x-amz-checksum-algorithm`	Required headers: `x-amz-checksum-crc32`	Required headers: All part-level checksums need to be included in the CompleteM

Checksum algorithm	Checksum type	CreateMultipartUpload	UploadPart	CompleteMultipartUpload
MD5 (MD5) XXHash64 (XXHASH64) XXHash3 (XXHASH3) XXHash128 (XXHASH128) SHA-512 (SHA512)			x-amz-checksum-crc32c x-amz-checksum-sha1 x-amz-checksum-sha256 x-amz-checksum-md5 x-amz-checksum-xxhash64 x-amz-checksum-xxhash3 x-amz-checksum-xxhash128 x-amz-checksum-sha512	ultiPartUpload request. Optional headers: x-amz-checksum-crc32 x-amz-checksum-crc32c x-amz-checksum-sha1 x-amz-checksum-sha256 x-amz-checksum-md5 x-amz-checksum-xxhash64 x-amz-checksum-xxhash3 x-amz-checksum-xxhash128

Checksum algorithm	Checksum type	CreateMultipartUpload	UploadPart	CompleteMultipartUpload
				x-amz-checksum-sha512

Topics

- Configuring a bucket lifecycle configuration to delete incomplete multipart uploads
- Uploading an object using multipart upload
- Uploading a directory using the high-level .NET TransferUtility class
- Listing multipart uploads
- Tracking a multipart upload with the AWS SDKs
- Aborting a multipart upload
- Copying an object using multipart upload
- Tutorial: Upload an object through multipart upload and verify its data integrity
- Amazon S3 multipart upload limits

Configuring a bucket lifecycle configuration to delete incomplete multipart uploads

As a best practice, we recommend that you configure a lifecycle rule by using the `AbortIncompleteMultipartUpload` action to minimize your storage costs. For more information about aborting a multipart upload, see Aborting a multipart upload.

Amazon S3 supports a bucket lifecycle rule that you can use to direct Amazon S3 to stop multipart uploads that aren't completed within a specified number of days after being initiated. When a multipart upload isn't completed within the specified time frame, it becomes eligible for an abort operation. Amazon S3 then stops the multipart upload and deletes the parts associated with the multipart upload. This rule applies to both existing multipart uploads and those that you create later.

The following is an example lifecycle configuration that specifies a rule with the `AbortIncompleteMultipartUpload` action.

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Prefix></Prefix>
        <Status>Enabled</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

In the example, the rule doesn't specify a value for the `Prefix` element (the object key name prefix). Therefore, the rule applies to all objects in the bucket for which you initiated multipart uploads. Any multipart uploads that were initiated and weren't completed within seven days become eligible for an abort operation. The abort action has no effect on completed multipart uploads.

For more information about the bucket lifecycle configuration, see Managing the lifecycle of objects.

> **Note**
>
> If the multipart upload is completed within the number of days specified in the rule, the `AbortIncompleteMultipartUpload` lifecycle action does not apply (that is, Amazon S3 doesn't take any action). Also, this action doesn't apply to objects. No objects are deleted by this lifecycle action. Additionally, you will not incur early delete charges for S3 Lifecycle when you remove any incomplete multipart upload parts.

Using the S3 console

To automatically manage incomplete multipart uploads, you can use the S3 console to create a lifecycle rule to expire incomplete multipart upload bytes from your bucket after a specified number of days. The following procedure shows you how to add a lifecycle rule to delete incomplete multipart uploads after 7 days. For more information about adding lifecycle rules, see Setting an S3 Lifecycle configuration on a bucket.

To add a lifecycle rule to abort incomplete multipart uploads that are more than 7 days old

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to create a lifecycle rule for.
3. Choose the **Management** tab, and choose **Create lifecycle rule**.
4. In **Lifecycle rule name**, enter a name for your rule.

   The name must be unique within the bucket.
5. Choose the scope of the lifecycle rule:

   - To create a lifecycle rule for all objects with a specific prefix, choose **Limit the scope of this rule using one or more filters**, and enter the prefix in the **Prefix** field.
   - To create a lifecycle rule for all objects in the bucket, choose **This rule applies to all objects in the bucket**, and choose **I acknowledge that this rule applies to all objects in the bucket**.
6. Under **Lifecycle rule actions**, select **Delete expired object delete markers or incomplete multipart uploads**.
7. Under **Delete expired object delete markers or incomplete multipart uploads**, select **Delete incomplete multipart uploads**.
8. In the **Number of days** field, enter the number of days after which to delete incomplete multipart uploads (for this example, 7 days).
9. Choose **Create rule**.

Using the AWS CLI

The following `put-bucket-lifecycle-configuration` AWS Command Line Interface (AWS CLI) command adds the lifecycle configuration for the specified bucket. To use this command, replace the *`user input placeholders`* with your information.

```
aws s3api put-bucket-lifecycle-configuration  \
        --bucket amzn-s3-demo-bucket  \
        --lifecycle-configuration filename-containing-lifecycle-configuration
```

The following example shows how to add a lifecycle rule to abort incomplete multipart uploads by using the AWS CLI. It includes an example JSON lifecycle configuration to abort incomplete multipart uploads that are more than 7 days old.

To use the CLI commands in this example, replace the *user input placeholders* with your information.

To add a lifecycle rule to abort incomplete multipart uploads

1. Set up the AWS CLI. For instructions, see Developing with Amazon S3 using the AWS CLI in the *Amazon S3 API Reference*.
2. Save the following example lifecycle configuration in a file (for example, *lifecycle.json*). This example configuration specifies an empty prefix, and therefore it applies to all objects in the bucket. To restrict the configuration to a subset of objects, you can specify a prefix.

```
{
    "Rules": [
        {
            "ID": "Test Rule",
            "Status": "Enabled",
            "Filter": {
                "Prefix": ""
            },
            "AbortIncompleteMultipartUpload": {
                "DaysAfterInitiation": 7
            }
        }
    ]
}
```

3. Run the following CLI command to set this lifecycle configuration on your bucket.

```
aws s3api put-bucket-lifecycle-configuration   \
--bucket amzn-s3-demo-bucket  \
--lifecycle-configuration file://lifecycle.json
```

4. To verify that the lifecycle configuration has been set on your bucket, retrieve the lifecycle configuration by using the following `get-bucket-lifecycle` command.

```
aws s3api get-bucket-lifecycle  \
--bucket amzn-s3-demo-bucket
```

5. To delete the lifecycle configuration, use the following `delete-bucket-lifecycle` command.

```
aws s3api delete-bucket-lifecycle \
--bucket amzn-s3-demo-bucket
```

Uploading an object using multipart upload

You can use the multipart upload to programmatically upload a single object to Amazon S3. Each object is uploaded as a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently and in any order. If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object. Anonymous users cannot initiate multipart uploads.

> **Note**
>
> Multipart upload copy operations do not copy annotations from the source object by default. To include annotations when copying objects using multipart upload (for example, with the AWS CLI `aws s3 cp` command for objects larger than approximately 8 MB), specify `--copy-props all`.

For an end-to-end procedure on uploading an object with multipart upload with an additional checksum, see Tutorial: Upload an object through multipart upload and verify its data integrity.

The following section show how to use multipart upload with the AWS Command Line Interface, and AWS SDKs.

Using the S3 console

You can upload any file type—images, backups, data, movies, and so on—into an S3 bucket. The maximum size of a file that you can upload by using the Amazon S3 console is 160 GB. To upload a file larger than 160 GB, use the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

For instructions on uploading an object through the AWS Management Console, see Uploading objects.

Using the AWS CLI

The following describe the Amazon S3 operations for multipart upload using the AWS CLI.

- Initiate Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- Abort Multipart Upload
- List Parts
- List Multipart Uploads

Using the REST API

The following sections in the *Amazon Simple Storage Service API Reference* describe the REST API for multipart upload.

- Initiate Multipart Upload
- Upload Part
- Complete Multipart Upload
- Stop Multipart Upload
- List Parts
- List Multipart Uploads

Using the AWS SDKs (high-level API)

Some AWS SDKs expose a high-level API that simplifies multipart upload by combining the different API operations required to complete a multipart upload into a single operation. For more information, see Uploading and copying objects using multipart upload in Amazon S3.

If you need to pause and resume multipart uploads, vary part sizes during the upload, or do not know the size of the data in advance, use the low-level API methods. The low-level API methods for multipart uploads offer additional functionality, for more information, see Using the AWS SDKs (low-level API).

Java

For examples of how to perform a multipart upload with the AWS SDK for Java, see Upload or download large files to and from Amazon S3 using an AWS SDK in the *Amazon S3 API Reference*.

.NET

To upload a file to an S3 bucket, use the `TransferUtility` class. When uploading data from a file, you must provide the object's key name. If you don't, the API uses the file name for the key name. When uploading data from a stream, you must provide the object's key name.

To set advanced upload options—such as the part size, the number of threads when uploading the parts concurrently, metadata, the storage class, or ACL—use the `TransferUtilityUploadRequest` class.

> **Note**
>
> When you're using a stream for the source of data, the `TransferUtility` class does not do concurrent uploads.

The following C# example uploads a file to an Amazon S3 bucket in multiple parts. It shows how to use various `TransferUtility.Upload` overloads to upload a file. Each successive call to upload replaces the previous upload. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPUHighLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object
 ***";
        private const string filePath = "*** provide the full path name of the file
 to upload ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
```

```
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadFileAsync().Wait();
        }

        private static async Task UploadFileAsync()
        {
            try
            {
                var fileTransferUtility =
                    new TransferUtility(s3Client);

                // Option 1. Upload a file. The file name is used as the object key
name.
                await fileTransferUtility.UploadAsync(filePath, bucketName);
                Console.WriteLine("Upload 1 completed");

                // Option 2. Specify object key name explicitly.
                await fileTransferUtility.UploadAsync(filePath, bucketName,
keyName);
                Console.WriteLine("Upload 2 completed");

                // Option 3. Upload data from a type of System.IO.Stream.
                using (var fileToUpload =
                    new FileStream(filePath, FileMode.Open, FileAccess.Read))
                {
                    await fileTransferUtility.UploadAsync(fileToUpload,
                                                bucketName, keyName);
                }
                Console.WriteLine("Upload 3 completed");

                // Option 4. Specify advanced settings.
                var fileTransferUtilityRequest = new TransferUtilityUploadRequest
                {
                    BucketName = bucketName,
                    FilePath = filePath,
                    StorageClass = S3StorageClass.StandardInfrequentAccess,
                    PartSize = 6291456, // 6 MB.
                    Key = keyName,
                    CannedACL = S3CannedACL.PublicRead
                };
```

```
                fileTransferUtilityRequest.Metadata.Add("param1", "Value1");
                fileTransferUtilityRequest.Metadata.Add("param2", "Value2");

                await fileTransferUtility.UploadAsync(fileTransferUtilityRequest);
                Console.WriteLine("Upload 4 completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }

        }
    }
}
```

JavaScript

Example

Upload a large file.

```
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

import {
  ProgressBar,
  logger,
} from "@aws-doc-sdk-examples/lib/utils/util-log.js";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

/**
 * Create a 25MB file and upload it in parts to the specified
 * Amazon S3 bucket.
```

```
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const str = createString();
  const buffer = Buffer.from(str, "utf8");
  const progressBar = new ProgressBar({
    description: `Uploading "${key}" to "${bucketName}"`,
    barLength: 30,
  });

  try {
    const upload = new Upload({
      client: new S3Client({}),
      params: {
        Bucket: bucketName,
        Key: key,
        Body: buffer,
      },
    });

    upload.on("httpUploadProgress", ({ loaded, total }) => {
      progressBar.update({ current: loaded, total });
    });

    await upload.done();
  } catch (caught) {
    if (caught instanceof Error && caught.name === "AbortError") {
      logger.error(`Multipart upload was aborted. ${caught.message}`);
    } else {
      throw caught;
    }
  }
};
```

Example

Download a large file.

```
import { fileURLToPath } from "node:url";
import { GetObjectCommand, NoSuchKey, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream, rmSync } from "node:fs";

const s3Client = new S3Client({});
```

```
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({
    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: Number.parseInt(start),
    end: Number.parseInt(end),
    length: Number.parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });
    console.log(`Downloaded bytes ${nextRange.start} to ${nextRange.end}`);
```

```
    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
};

/**
 * Download a large object from and Amazon S3 bucket.
 *
 * When downloading a large file, you might want to break it down into
 * smaller pieces. Amazon S3 accepts a Range header to specify the start
 * and end of the byte range to be downloaded.
 *
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  try {
    await downloadInChunks({
      bucket: bucketName,
      key: key,
    });
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(`Failed to download object. No such key "${key}".`);
      rmSync(key);
    }
  }
};
```

Go

For more information about the Go code example for multipart upload, see Upload or download large files to and from Amazon S3 using an AWS SDK.

Example

Upload a large object by using an upload manager to break the data into parts and upload them concurrently.

```
import (
 "bytes"
 "context"
```

```
 "errors"
 "fmt"
 "io"
 "log"
 "os"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/s3/types"
 "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
 S3Client *s3.Client
}
```

```
// UploadLargeObject uses an upload manager to upload data to an object in a bucket.
// The upload manager breaks large data into parts and uploads the parts
 concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName string,
 objectKey string, largeObject []byte) error {
 largeBuffer := bytes.NewReader(largeObject)
 var partMiBs int64 = 10
 uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
  u.PartSize = partMiBs * 1024 * 1024
 })
 _, err := uploader.Upload(ctx, &s3.PutObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
  Body:   largeBuffer,
 })
 if err != nil {
  var apiErr smithy.APIError
  if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
   log.Printf("Error while uploading object to %s. The object is too large.\n"+
```

```
    "The maximum size for a multipart upload is 5TB.", bucketName)
  } else {
   log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
    bucketName, objectKey, err)
  }
 } else {
  err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
   ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
 aws.String(objectKey)}, time.Minute)
  if err != nil {
   log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
 }

 return err
}
```

Example

Download a large object by using a download manager to get the data in parts and download them concurrently.

```
// DownloadLargeObject uses a download manager to download an object from a bucket.
// The download manager gets the data in parts and writes them to a buffer until all
 of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
 string, objectKey string) ([]byte, error) {
 var partMiBs int64 = 10
 downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
  d.PartSize = partMiBs * 1024 * 1024
 })
 buffer := manager.NewWriteAtBuffer([]byte{})
 _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
  Bucket: aws.String(bucketName),
  Key:    aws.String(objectKey),
 })
 if err != nil {
  log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
   bucketName, objectKey, err)
 }
```

```
 return buffer.Bytes(), err
}
```

PHP

This topic explains how to use the high-level `Aws\S3\Model\MultipartUpload\UploadBuilder` class from the AWS SDK for PHP for multipart file uploads. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

The following PHP example uploads a file to an Amazon S3 bucket. The example demonstrates how to set parameters for the `MultipartUploader` object.

```
 require 'vendor/autoload.php';

use Aws\Exception\MultipartUploadException;
use Aws\S3\MultipartUploader;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Prepare the upload parameters.
$uploader = new MultipartUploader($s3, '/path/to/large/file.zip', [
    'bucket' => $bucket,
    'key'    => $keyname
]);

// Perform the upload.
try {
    $result = $uploader->upload();
    echo "Upload complete: {$result['ObjectURL']}" . PHP_EOL;
} catch (MultipartUploadException $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Python

The following example loads an object using the high-level multipart upload Python API (the `TransferManager` class).

```
import sys
import threading

import boto3
from boto3.s3.transfer import TransferConfig


MB = 1024 * 1024
s3 = boto3.resource("s3")


class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
```

```
            else:
                self.thread_info[thread.ident] += bytes_transferred

            target = self._target_size * MB
            sys.stdout.write(
                f"\r{self._total_transferred} of {target} transferred "
                f"({(self._total_transferred / target) * 100:.2f}%)."
            )
            sys.stdout.flush()


def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
```

```
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )
    return transfer_callback.thread_info


def upload_with_high_threshold(local_file_path, bucket_name, object_key,
 file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
```

```
        local_file_path, object_key, ExtraArgs=extra_args,
 Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
```

```
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info


def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey": sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

Using the AWS SDKs (low-level API)

The AWS SDK exposes a low-level API that closely resembles the Amazon S3 REST API for multipart uploads (see Uploading and copying objects using multipart upload in Amazon S3. Use the low-level API when you need to pause and resume multipart uploads, vary part sizes during the upload, or do not know the size of the upload data in advance. When you don't have these requirements, use the high-level API (see Using the AWS SDKs (high-level API)).

Java

The following example shows how to use the low-level Java classes to upload a file. It performs the following steps:

- Initiates a multipart upload using the `AmazonS3Client.initiateMultipartUpload()` method, and passes in an `InitiateMultipartUploadRequest` object.
- Saves the upload ID that the `AmazonS3Client.initiateMultipartUpload()` method returns. You provide this upload ID for each subsequent multipart upload operation.
- Uploads the parts of the object. For each part, you call the `AmazonS3Client.uploadPart()` method. You provide part upload information using an `UploadPartRequest` object.
- For each part, saves the ETag from the response of the `AmazonS3Client.uploadPart()` method in a list. You use the ETag values to complete the multipart upload.
- Calls the `AmazonS3Client.completeMultipartUpload()` method to complete the multipart upload.

Example

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class LowLevelMultipartUpload {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
```

```
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String filePath = "*** Path to file to upload ***";

        File file = new File(filePath);
        long contentLength = file.length();
        long partSize = 5 * 1024 * 1024; // Set part size to 5 MB.

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Create a list of ETag objects. You retrieve ETags for each object
part
            // uploaded,
            // then, after each individual part has been uploaded, pass the list of
ETags to
            // the request to complete the upload.
            List<PartETag> partETags = new ArrayList<PartETag>();

            // Initiate the multipart upload.
            InitiateMultipartUploadRequest initRequest = new
InitiateMultipartUploadRequest(bucketName, keyName);
            InitiateMultipartUploadResult initResponse =
s3Client.initiateMultipartUpload(initRequest);

            // Upload the file parts.
            long filePosition = 0;
            for (int i = 1; filePosition < contentLength; i++) {
                // Because the last part could be less than 5 MB, adjust the part
size as
                // needed.
                partSize = Math.min(partSize, (contentLength - filePosition));

                // Create the request to upload a part.
                UploadPartRequest uploadRequest = new UploadPartRequest()
                        .withBucketName(bucketName)
                        .withKey(keyName)
                        .withUploadId(initResponse.getUploadId())
                        .withPartNumber(i)
                        .withFileOffset(filePosition)
                        .withFile(file)
```

```
                        .withPartSize(partSize);

                // Upload the part and add the response's ETag to our list.
                UploadPartResult uploadResult = s3Client.uploadPart(uploadRequest);
                partETags.add(uploadResult.getPartETag());

                filePosition += partSize;
            }

            // Complete the multipart upload.
            CompleteMultipartUploadRequest compRequest = new
 CompleteMultipartUploadRequest(bucketName, keyName,
                    initResponse.getUploadId(), partETags);
            s3Client.completeMultipartUpload(compRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

The following C# example shows how to use the low-level SDK for .NET multipart upload API to upload a file to an S3 bucket. For information about Amazon S3 multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3.

> **Note**
>
> When you use the SDK for .NET API to upload large objects, a timeout might occur while data is being written to the request stream. You can set an explicit timeout using the `UploadPartRequest`.

The following C# example uploads a file to an S3 bucket using the low-level multipart upload API. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadFileMPULowLevelAPITest
    {
        private const string bucketName = "*** provide bucket name ***";
        private const string keyName = "*** provide a name for the uploaded object
 ***";
        private const string filePath = "*** provide the full path name of the file
 to upload ***";
       // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Uploading an object");
            UploadObjectAsync().Wait();
        }

        private static async Task UploadObjectAsync()
        {
            // Create list to store upload part responses.
            List<UploadPartResponse> uploadResponses = new
 List<UploadPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest = new
 InitiateMultipartUploadRequest
```

```
            {
                BucketName = bucketName,
                Key = keyName
            };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // Upload parts.
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                Console.WriteLine("Uploading parts");

                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                        {
                            BucketName = bucketName,
                            Key = keyName,
                            UploadId = initResponse.UploadId,
                            PartNumber = i,
                            PartSize = partSize,
                            FilePosition = filePosition,
                            FilePath = filePath
                        };

                    // Track upload progress.
                    uploadRequest.StreamTransferProgress +=
                        new
EventHandler<StreamTransferProgressArgs>(UploadPartProgressEventCallback);

                    // Upload a part and add the response to our list.
                    uploadResponses.Add(await
s3Client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                // Setup to complete the upload.
```

```
                CompleteMultipartUploadRequest completeRequest = new
 CompleteMultipartUploadRequest
                    {
                        BucketName = bucketName,
                        Key = keyName,
                        UploadId = initResponse.UploadId
                     };
                completeRequest.AddPartETags(uploadResponses);

                // Complete the upload.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown: { 0}",
 exception.Message);

                // Abort the upload.
                AbortMultipartUploadRequest abortMPURequest = new
 AbortMultipartUploadRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    UploadId = initResponse.UploadId
                };
               await s3Client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
        public static void UploadPartProgressEventCallback(object sender,
 StreamTransferProgressArgs e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

PHP

This topic shows how to use the low-level `uploadPart` method from version 3 of the AWS SDK for PHP to upload a file in multiple parts. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

The following PHP example uploads a file to an Amazon S3 bucket using the low-level PHP API multipart upload.

```
 require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$filename = '*** Path to and Name of the File to Upload ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

$result = $s3->createMultipartUpload([
    'Bucket'       => $bucket,
    'Key'          => $keyname,
    'StorageClass' => 'REDUCED_REDUNDANCY',
    'Metadata'     => [
        'param1' => 'value 1',
        'param2' => 'value 2',
        'param3' => 'value 3'
    ]
]);
$uploadId = $result['UploadId'];

// Upload the file in parts.
try {
    $file = fopen($filename, 'r');
    $partNumber = 1;
    while (!feof($file)) {
        $result = $s3->uploadPart([
            'Bucket'     => $bucket,
            'Key'        => $keyname,
            'UploadId'   => $uploadId,
            'PartNumber' => $partNumber,
            'Body'       => fread($file, 5 * 1024 * 1024),
        ]);
        $parts['Parts'][$partNumber] = [
            'PartNumber' => $partNumber,
```

```
            'ETag' => $result['ETag'],
        ];
        $partNumber++;

        echo "Uploading part $partNumber of $filename." . PHP_EOL;
    }
    fclose($file);
} catch (S3Exception $e) {
    $result = $s3->abortMultipartUpload([
        'Bucket'   => $bucket,
        'Key'      => $keyname,
        'UploadId' => $uploadId
    ]);

    echo "Upload of $filename failed." . PHP_EOL;
}

// Complete the multipart upload.
$result = $s3->completeMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
    'UploadId' => $uploadId,
    'MultipartUpload'    => $parts,
]);
$url = $result['Location'];

echo "Uploaded $filename to $url." . PHP_EOL;
```

Using the AWS SDK for Ruby

The AWS SDK for Ruby version 3 supports Amazon S3 multipart uploads in two ways. For the first option, you can use managed file uploads. For more information, see Uploading Files to Amazon S3 in the *AWS Developer Blog*. Managed file uploads are the recommended method for uploading files to a bucket. They provide the following benefits:

- Manage multipart uploads for objects larger than 15MB.
- Correctly open files in binary mode to avoid encoding issues.
- Use multiple threads for uploading parts of large objects in parallel.

Alternatively, you can use the following multipart upload client operations directly:

- create_multipart_upload – Initiates a multipart upload and returns an upload ID.
- upload_part – Uploads a part in a multipart upload.
- upload_part_copy – Uploads a part by copying data from an existing object as data source.
- complete_multipart_upload – Completes a multipart upload by assembling previously uploaded parts.
- abort_multipart_upload – Stops a multipart upload.

Uploading a directory using the high-level .NET TransferUtility class

You can use the `TransferUtility` class to upload an entire directory. By default, the API uploads only the files at the root of the specified directory. You can, however, specify recursively uploading files in all of the sub directories.

To select files in the specified directory based on filtering criteria, specify filtering expressions. For example, to upload only the PDF files from a directory, specify the `"*.pdf"` filter expression.

When uploading files from a directory, you don't specify the key names for the resulting objects. Amazon S3 constructs the key names using the original file path. For example, assume that you have a directory called `c:\myfolder` with the following structure:

Example

```
C:\myfolder
      \a.txt
      \b.pdf
      \media\
             An.mp3
```

When you upload this directory, Amazon S3 uses the following key names:

Example

```
a.txt
b.pdf
media/An.mp3
```

Example

The following C# example uploads a directory to an Amazon S3 bucket. It shows how to use various `TransferUtility.UploadDirectory` overloads to upload the directory. Each successive call to upload replaces the previous upload. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.IO;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class UploadDirMPUHighLevelAPITest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string directoryPath = @"*** directory path ***";
        // The example uploads only .txt files.
        private const string wildCard = "*.txt";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            UploadDirAsync().Wait();
        }

        private static async Task UploadDirAsync()
        {
            try
            {
                var directoryTransferUtility =
                    new TransferUtility(s3Client);

                // 1. Upload a directory.
                await directoryTransferUtility.UploadDirectoryAsync(directoryPath,
                    existingBucketName);
                Console.WriteLine("Upload statement 1 completed");
```

```
                // 2. Upload only the .txt files from a directory
                //    and search recursively.
                await directoryTransferUtility.UploadDirectoryAsync(
                                                directoryPath,
                                                existingBucketName,
                                                wildCard,
                                                SearchOption.AllDirectories);
                Console.WriteLine("Upload statement 2 completed");

                // 3. The same as Step 2 and some optional configuration.
                //    Search recursively for .txt files to upload.
                var request = new TransferUtilityUploadDirectoryRequest
                {
                    BucketName = existingBucketName,
                    Directory = directoryPath,
                    SearchOption = SearchOption.AllDirectories,
                    SearchPattern = wildCard
                };

                await directoryTransferUtility.UploadDirectoryAsync(request);
                Console.WriteLine("Upload statement 3 completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine(
                        "Error encountered ***. Message:'{0}' when writing an object",
 e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine(
                    "Unknown encountered on server. Message:'{0}' when writing an
 object", e.Message);
            }
        }
    }
}
```

Listing multipart uploads

You can use the AWS CLI, REST API, or AWS SDKs, to retrieve a list of in-progress multipart uploads in Amazon S3. You can use the multipart upload to programmatically upload a single object to Amazon S3. Multipart uploads move objects into Amazon S3 by moving a portion of an object's

data at a time. For more general information about multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3.

For an end-to-end procedure on uploading an object with multipart upload with an additional checksum, see Tutorial: Upload an object through multipart upload and verify its data integrity.

The following section show how to list in-progress multipart uploads with the AWS Command Line Interface, the Amazon S3 REST API, and AWS SDKs.

Listing multipart uploads using the AWS CLI

The following sections in the AWS Command Line Interface describe the operations for listing multipart uploads.

- list-parts-list the uploaded parts for a specific multipart upload.
- list-multipart-uploads-list in-progress multipart uploads.

Listing multipart uploads using the REST API

The following sections in the *Amazon Simple Storage Service API Reference* describe the REST API for listing multipart uploads:

- ListParts-list the uploaded parts for a specific multipart upload.
- ListMultipartUploads-list in-progress multipart uploads.

Listing multipart uploads using the AWS SDK (low-level API)

Java

To list all in-progress multipart uploads on a bucket using the AWS SDK for Java, you can use the low-level API classes to:

Low-level API multipart uploads listing process

1	Create an instance of the `ListMultipartUploadsRequest` class and provide the bucket name.
2	Run the S3Client `listMultipartUploads` method. The method returns an instance of the `ListMultipartUploadsResponse` class that gives you information about the multipart uploads in progress.

For examples of how to list multipart uploads with the AWS SDK for Java, see List multipart uploads in the *Amazon S3 API Reference*.

.NET

To list all of the in-progress multipart uploads on a specific bucket, use the SDK for .NET low-level multipart upload API's `ListMultipartUploadsRequest` class. The `AmazonS3Client.ListMultipartUploads` method returns an instance of the `ListMultipartUploadsResponse` class that provides information about the in-progress multipart uploads.

An in-progress multipart upload is a multipart upload that has been initiated using the initiate multipart upload request, but has not yet been completed or stopped. For more information about Amazon S3 multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3.

The following C# example shows how to use the SDK for .NET to list all in-progress multipart uploads on a bucket. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
ListMultipartUploadsRequest request = new ListMultipartUploadsRequest
{
  BucketName = bucketName // Bucket receiving the uploads.
};

ListMultipartUploadsResponse response = await
 AmazonS3Client.ListMultipartUploadsAsync(request);
```

PHP

This topic shows how to use the low-level API classes from version 3 of the AWS SDK for PHP to list all in-progress multipart uploads on a bucket. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

The following PHP example demonstrates listing all in-progress multipart uploads on a bucket.

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
```

```
$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Retrieve a list of the current multipart uploads.
$result = $s3->listMultipartUploads([
    'Bucket' => $bucket
]);

// Write the list of uploads to the page.
print_r($result->toArray());
```

Tracking a multipart upload with the AWS SDKs

You can track an object's upload progress to Amazon S3 with a listen interface. The high-level multipart upload API provides such a listen interface, called `ProgressListener`. Progress events occur periodically and notify the listener that bytes have been transferred. For more general information about multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3.

For an end-to-end procedure on uploading an object with multipart upload with an additional checksum, see Tutorial: Upload an object through multipart upload and verify its data integrity.

The following section show how to track a multipart upload with the AWS SDKs.

Java

Example

The following Java code uploads a file and uses the `ExecutionInterceptor` to track the upload progress. For instructions on how to create and test a working sample, see Getting Started in the AWS SDK for Java 2.x Developer Guide.

```
import java.nio.file.Paths;

import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.interceptor.Context;
import software.amazon.awssdk.core.interceptor.ExecutionAttributes;
import software.amazon.awssdk.core.interceptor.ExecutionInterceptor;
```

```
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;

public class TrackMPUProgressUsingHighLevelAPI {

    static class ProgressListener implements ExecutionInterceptor {
        private long transferredBytes = 0;

        @Override
        public void beforeTransmission(Context.BeforeTransmission context,
 ExecutionAttributes executionAttributes) {
            if (context.httpRequest().firstMatchingHeader("Content-
Length").isPresent()) {
                String contentLength =
 context.httpRequest().firstMatchingHeader("Content-Length").get();
                long partSize = Long.parseLong(contentLength);
                transferredBytes += partSize;
                System.out.println("Transferred bytes: " + transferredBytes);
            }
        }
    }

    public static void main(String[] args) throws Exception {
        String existingBucketName = "*** Provide bucket name ***";
        String keyName = "*** Provide object key ***";
        String filePath = "*** file to upload ***";

        S3AsyncClient s3Client = S3AsyncClient.builder()
                .credentialsProvider(ProfileCredentialsProvider.create())
                .overrideConfiguration(c -> c.addExecutionInterceptor(new
 ProgressListener()))
                .build();

        // For more advanced uploads, you can create a request object
        // and supply additional request parameters (ex: progress listeners,
        // canned ACLs, etc.)
        PutObjectRequest request = PutObjectRequest.builder()
                .bucket(existingBucketName)
                .key(keyName)
                .build();

        AsyncRequestBody requestBody =
 AsyncRequestBody.fromFile(Paths.get(filePath));
```

```
        // You can ask the upload for its progress, or you can
        // add a ProgressListener to your request to receive notifications
        // when bytes are transferred.
        // S3AsyncClient processes all transfers asynchronously,
        // so this call will return immediately.
        var upload = s3Client.putObject(request, requestBody);

        try {
            // You can block and wait for the upload to finish
            upload.join();
        } catch (Exception exception) {
            System.out.println("Unable to upload file, upload aborted.");
            exception.printStackTrace();
        } finally {
            s3Client.close();
        }
    }
}
```

.NET

The following C# example uploads a file to an S3 bucket using the `TransferUtility` class, and tracks the progress of the upload. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class TrackMPUUsingHighLevelAPITest
    {
        private const string bucketName = "*** provide the bucket name ***";
        private const string keyName = "*** provide the name for the uploaded object
 ***";
        private const string filePath = " *** provide the full path name of the file
 to upload **";
        // Specify your bucket region (an example region is shown).
```

```
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;


        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            TrackMPUAsync().Wait();
        }

        private static async Task TrackMPUAsync()
        {
            try
            {
                var fileTransferUtility = new TransferUtility(s3Client);

                // Use TransferUtilityUploadRequest to configure options.
                // In this example we subscribe to an event.
                var uploadRequest =
                    new TransferUtilityUploadRequest
                    {
                        BucketName = bucketName,
                        FilePath = filePath,
                        Key = keyName
                    };

                uploadRequest.UploadProgressEvent +=
                    new EventHandler<UploadProgressArgs>
                        (uploadRequest_UploadPartProgressEvent);

                await fileTransferUtility.UploadAsync(uploadRequest);
                Console.WriteLine("Upload completed");
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
```

```
        }

        static void uploadRequest_UploadPartProgressEvent(object sender,
 UploadProgressArgs e)
        {
            // Process event.
            Console.WriteLine("{0}/{1}", e.TransferredBytes, e.TotalBytes);
        }
    }
}
```

Aborting a multipart upload

After you initiate a multipart upload, you begin uploading parts. Amazon S3 stores these parts, and only creates the object after you upload all parts and send a request to complete the multipart upload. Upon receiving the complete multipart upload request, Amazon S3 assembles the parts and creates an object. If you don't send the complete multipart upload request successfully, S3 does not assemble the parts and does not create any object. If you wish to not complete a multipart upload after uploading parts you should abort the multipart upload.

You are billed for all storage associated with uploaded parts. It's recommended to always either complete the multipart upload or stop the multipart upload to remove any uploaded parts. For more information about pricing, see Multipart upload and pricing.

You can also stop an incomplete multipart upload using a bucket lifecycle configuration. For more information, see Configuring a bucket lifecycle configuration to delete incomplete multipart uploads.

The following section show how to stop an in-progress multipart upload in Amazon S3 using the AWS Command Line Interface, REST API, or AWS SDKs.

Using the AWS CLI

For more information about using the AWS CLI to stop a multipart upload, see abort-multipart-upload in the *AWS CLI Command Reference*.

Using the REST API

For more information about using the REST API to stop a multipart upload, see AbortMultipartUpload in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs (high-level API)

Java

To stop multipart uploads in progress using the AWS SDK for Java, you can abort uploads that were initiated before a specified date and are still in progress. An upload is considered to be in progress after you initiate it and until you complete it or stop it.

To stop multipart uploads, you can:

1	Create an S3Client instance.
2	Use the client's abort methods by passing the bucket name and other required parameters.

> **Note**
>
> You can also stop a specific multipart upload. For more information, see Using the AWS SDKs (low-level API).

For examples of how to abort multipart uploads with the AWS SDK for Java, see Cancel a multipart upload in the *Amazon S3 API Reference*.

.NET

The following C# example stops all in-progress multipart uploads that were initiated on a specific bucket over a week ago. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Transfer;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class AbortMPUUsingHighLevelAPITest
```

```
    {
        private const string bucketName = "*** provide bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            AbortMPUAsync().Wait();
        }

        private static async Task AbortMPUAsync()
        {
            try
            {
                var transferUtility = new TransferUtility(s3Client);

                // Abort all in-progress uploads initiated before the specified
 date.
                await transferUtility.AbortMultipartUploadsAsync(
                    bucketName, DateTime.Now.AddDays(-7));
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

Note

You can also stop a specific multipart upload. For more information, see Using the AWS SDKs (low-level API).

Using the AWS SDKs (low-level API)

You can stop an in-progress multipart upload by calling the `AmazonS3.abortMultipartUpload` method. This method deletes any parts that were uploaded to Amazon S3 and frees up the resources. You must provide the upload ID, bucket name, and key name. The following Java code example demonstrates how to stop an in-progress multipart upload.

To stop a multipart upload, you provide the upload ID, and the bucket and key names that are used in the upload. After you have stopped a multipart upload, you can't use the upload ID to upload additional parts. For more information about Amazon S3 multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3.

Java

To stop a specific in-progress multipart upload using the AWS SDK for Java, you can use the low-level API to abort the upload by providing the bucket name, object key, and upload ID.

> **Note**
>
> Instead of aborting a specific multipart upload, you can stop all multipart uploads initiated before a specific time that are still in progress. This clean-up operation is useful to stop old multipart uploads that you initiated but did not complete or stop. For more information, see Using the AWS SDKs (high-level API).

For examples of how to abort a specific multipart upload with the AWS SDK for Java, see Cancel a multipart upload in the *Amazon S3 API Reference*.

.NET

The following C# example shows how to stop a multipart upload. For a complete C# sample that includes the following code, see Using the AWS SDKs (low-level API).

```
AbortMultipartUploadRequest abortMPURequest = new AbortMultipartUploadRequest
{
    BucketName = existingBucketName,
    Key = keyName,
    UploadId = initResponse.UploadId
};
await AmazonS3Client.AbortMultipartUploadAsync(abortMPURequest);
```

You can also abort all in-progress multipart uploads that were initiated prior to a specific time. This clean-up operation is useful for aborting multipart uploads that didn't complete or were aborted. For more information, see Using the AWS SDKs (high-level API).

PHP

This example shows how to use a class from version 3 of the AWS SDK for PHP to abort a multipart upload that is in progress. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2. The example the `abortMultipartUpload()` method.

For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';
$uploadId = '*** Upload ID of upload to Abort ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Abort the multipart upload.
$s3->abortMultipartUpload([
    'Bucket'   => $bucket,
    'Key'      => $keyname,
    'UploadId' => $uploadId,
]);
```

Copying an object using multipart upload

Multipart upload allows you to copy objects as a set of parts. The examples in this section show you how to copy objects greater than 5 GB using the multipart upload API. For information about multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3.

You can copy objects less than 5 GB in a single operation without using the multipart upload API. You can copy objects less than 5 GB using the AWS Management Console, AWS CLI, REST API, or AWS SDKs. For more information, see Copying, moving, and renaming objects.

For an end-to-end procedure on uploading an object with multipart upload with an additional checksum, see Tutorial: Upload an object through multipart upload and verify its data integrity.

The following section show how to copy an object with multipart upload with the REST API or AWS SDKs.

Using the REST API

The following sections in the *Amazon Simple Storage Service API Reference* describe the REST API for multipart upload. For copying an existing object, use the Upload Part (Copy) API and specify the source object by adding the `x-amz-copy-source` request header in your request.

- Initiate Multipart Upload
- Upload Part
- Upload Part (Copy)
- Complete Multipart Upload
- Abort Multipart Upload
- List Parts
- List Multipart Uploads

You can use these APIs to make your own REST requests, or you can use one of the SDKs we provide. For more information about using Multipart Upload with the AWS CLI, see Using the AWS CLI. For more information about the SDKs, see AWS SDK support for multipart upload.

Using the AWS SDKs

To copy an object using the low-level API, do the following:

- Initiate a multipart upload by calling the `AmazonS3Client.initiateMultipartUpload()` method.
- Save the upload ID from the response object that the `AmazonS3Client.initiateMultipartUpload()` method returns. You provide this upload ID for each part-upload operation.
- Copy all of the parts. For each part that you need to copy, create a new instance of the `CopyPartRequest` class. Provide the part information, including the source and destination bucket names, source and destination object keys, upload ID, locations of the first and last bytes of the part, and part number.

- Save the responses of the `AmazonS3Client.copyPart()` method calls. Each response includes the `ETag` value and part number for the uploaded part. You need this information to complete the multipart upload.
- Call the `AmazonS3Client.completeMultipartUpload()` method to complete the copy operation.

Java

For examples of how to copy objects using multipart upload with the AWS SDK for Java, see Copy part of an object from another object in the *Amazon S3 API Reference*.

.NET

The following C# example shows how to use the SDK for .NET to copy an Amazon S3 object that is larger than 5 GB from one source location to another, such as from one bucket to another. To copy objects that are smaller than 5 GB, use the single-operation copy procedure described in Using the AWS SDKs. For more information about Amazon S3 multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3.

This example shows how to copy an Amazon S3 object that is larger than 5 GB from one S3 bucket to another using the SDK for .NET multipart upload API.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CopyObjectUsingMPUapiTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with
 source object ***";
        private const string targetBucket = "*** provide the name of the bucket to
 copy the object to ***";
        private const string sourceObjectKey = "*** provide the name of object to
 copy ***";
        private const string targetObjectKey = "*** provide the name of the object
 copy ***";
        // Specify your bucket region (an example region is shown).
```

```
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            MPUCopyObjectAsync().Wait();
        }
        private static async Task MPUCopyObjectAsync()
        {
            // Create a list to store the upload part responses.
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();
            List<CopyPartResponse> copyResponses = new List<CopyPartResponse>();

            // Setup information required to initiate the multipart upload.
            InitiateMultipartUploadRequest initiateRequest =
                new InitiateMultipartUploadRequest
                {
                    BucketName = targetBucket,
                    Key = targetObjectKey
                };

            // Initiate the upload.
            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // Save the upload ID.
            String uploadId = initResponse.UploadId;

            try
            {
                // Get the size of the object.
                GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
                {
                    BucketName = sourceBucket,
                    Key = sourceObjectKey
                };

                GetObjectMetadataResponse metadataResponse =
                    await s3Client.GetObjectMetadataAsync(metadataRequest);
```

```
                long objectSize = metadataResponse.ContentLength; // Length in
bytes.

                // Copy the parts.
                long partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

                long bytePosition = 0;
                for (int i = 1; bytePosition < objectSize; i++)
                {
                    CopyPartRequest copyRequest = new CopyPartRequest
                    {
                        DestinationBucket = targetBucket,
                        DestinationKey = targetObjectKey,
                        SourceBucket = sourceBucket,
                        SourceKey = sourceObjectKey,
                        UploadId = uploadId,
                        FirstByte = bytePosition,
                        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
                        PartNumber = i
                    };

                    copyResponses.Add(await s3Client.CopyPartAsync(copyRequest));

                    bytePosition += partSize;
                }

                // Set up to complete the copy.
                CompleteMultipartUploadRequest completeRequest =
                new CompleteMultipartUploadRequest
                {
                    BucketName = targetBucket,
                    Key = targetObjectKey,
                    UploadId = initResponse.UploadId
                };
                completeRequest.AddPartETags(copyResponses);

                // Complete the copy.
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (AmazonS3Exception e)
            {
```

```
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

Tutorial: Upload an object through multipart upload and verify its data integrity

Multipart upload allows you to upload a single object as a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently and in any order. If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object. In general, when your object size reaches 100 MB, you should consider using multipart uploads instead of uploading the object in a single operation. For more information about multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3. For limits related to multipart uploads, see Amazon S3 multipart upload limits.

You can use checksums to verify that assets are not altered when they are copied. Performing a checksum consists of using an algorithm to iterate sequentially over every byte in a file. Amazon S3 offers multiple checksum options for checking the integrity of data. We recommend that you perform these integrity checks as a durability best practice and to confirm that every byte is transferred without alteration. Amazon S3 also supports the following algorithms: SHA-1, SHA-256, CRC32, and CRC32C. Amazon S3 uses one or more of these algorithms to compute an additional checksum value and store it as part of the object metadata. For more information about checksums, see Checking object integrity in Amazon S3.

Objective

In this tutorial, you will learn how to upload an object to Amazon S3 by using a multipart upload and an additional SHA-256 checksum through the AWS Command Line Interface (AWS CLI). You'll also learn how to check the object's data integrity by calculating the MD5 hash and SHA-256 checksum of the uploaded object.

Topics

- Prerequisites
- Step 1: Create a large file
- Step 2: Split the file into multiple files
- Step 3: Create the multipart upload with an additional checksum
- Step 4: Upload the parts of your multipart upload
- Step 5: List all the parts of your multipart upload
- Step 6: Complete the multipart upload
- Step 7: Confirm that the object is uploaded to your bucket
- Step 8: Verify object integrity with an MD5 checksum
- Step 9: Verify object integrity with an additional checksum
- Step 10: Clean up your resources

Prerequisites

- Before you start this tutorial, make sure that you have access to an Amazon S3 bucket that you can upload to. For more information, see Creating a general purpose bucket.
- You must have the AWS CLI installed and configured. If you don't have the AWS CLI installed, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.
- Alternatively, you can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? and Getting started with AWS CloudShell in the *AWS CloudShell User Guide*.

Step 1: Create a large file

If you already have a file ready for upload, you can use the file for this tutorial. Otherwise, create a 15 MB file using the following steps. For limits related to multipart uploads, see Amazon S3 multipart upload limits.

To create a large file

Use one of the following commands to create your file, depending on which operating system you're using.

Linux or macOS

To create a 15 MB file, open your local terminal and run the following command:

```
dd if=/dev/urandom of=census-data.bin bs=1M count=15
```

This command creates a file named `census-data.bin` filled with random bytes, with a size of 15 MB.

Windows

To create a 15 MB file, open your local terminal and run the following command:

```
fsutil file createnew census-data.bin 15728640
```

This command creates a file named `census-data.bin` with a size of 15 MB of arbitrary data (15728640 bytes).

Step 2: Split the file into multiple files

To perform the multipart upload, you have to split your large file into smaller parts. You can then upload the smaller parts by using the multipart upload process. This step demonstrates how to split the large file created in Step 1 into smaller parts. The following example uses a 15 MB file named `census-data.bin`.

To split a large file into parts

Linux or macOS

To divide the large file into 5 MB parts, use the `split` command. Open your terminal and run the following:

```
split -b 5M -d census-data.bin census-part
```

This command splits `census-data.bin` into 5 MB parts named `census-part**`, where `**` is a numeric suffix starting from `00`.

Windows

To split the large file, use PowerShell. Open Powershell, and run the following script:

```
$inputFile = "census-data.bin"
$outputFilePrefix = "census-part"
```

```
$chunkSize = 5MB

$fs = [System.IO.File]::OpenRead($inputFile)
$buffer = New-Object byte[] $chunkSize
$fileNumber = 0

while ($fs.Position -lt $fs.Length) {
$bytesRead = $fs.Read($buffer, 0, $chunkSize)
$outputFile = "{0}{1:D2}" -f $outputFilePrefix, $fileNumber
$fileStream = [System.IO.File]::Create($outputFile)
$fileStream.Write($buffer, 0, $bytesRead)
$fileStream.Close()
$fileNumber++
}

$fs.Close()
```

This PowerShell script reads the large file in chunks of 5 MB and writes each chunk to a new file with a numeric suffix.

After running the appropriate command, you should see the parts in the directory where you executed the command. Each part will have a suffix corresponding to its part number, for example:

```
census-part00 census-part01 census-part02
```

Step 3: Create the multipart upload with an additional checksum

To begin the multipart upload process, you need to create the multipart upload request. This step involves initiating the multipart upload and specifying an additional checksum for data integrity. The following example uses the SHA-256 checksum. If you want to provide any metadata describing the object being uploaded, you must provide it in the request to initiate the multipart upload.

> **Note**
>
> In this step and subsequent steps, this tutorial uses the SHA-256 additional algorithm. You might optionally use another additional checksum for these steps, such as CRC32, CRC32C, or SHA-1. If you use a different algorithm, you must use it throughout the tutorial steps.

To start the multipart upload

In your terminal, use the following `create-multipart-upload` command to start a multipart upload for your bucket. Replace *amzn-s3-demo-bucket1* with your actual bucket name. Also, replace the `census_data_file` with your chosen file name. This file name becomes the object key when the upload completes.

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket1 --key
 'census_data_file' --checksum-algorithm sha256
```

If your request succeeds, you'll see JSON output like the following:

```
{
    "ServerSideEncryption": "AES256",
    "ChecksumAlgorithm": "SHA256",
    "Bucket": "amzn-s3-demo-bucket1",
    "Key": "census_data_file",
    "UploadId":
 "cNV6KCSNANFZapz1LUGPC5XwUVi1n6yUoIeSP138sNOKPeMhpKQRrbT9k0ePmgoOTCj9K83T4e2Gb5hQvNoNpCKqyb8m3
}
```

Note

When you send a request to initiate a multipart upload, Amazon S3 returns a response with an upload ID. This is a unique identifier for your multipart upload. You must include this upload ID whenever you upload parts, list the parts, complete an upload, or stop an upload. You'll need to use the `UploadId`, `Key`, and `Bucket` values for later steps, so make sure to save these.

Also, if you're using multipart upload with additional checksums, the part numbers must be consecutive. If you use nonconsecutive part numbers, the `complete-multipart-upload` request can result in an HTTP `500 Internal Server Error`.

Step 4: Upload the parts of your multipart upload

In this step, you will upload the parts of your multipart upload to your S3 bucket. Use the `upload-part` command to upload each part individually. This process requires specifying the upload ID, the part number, and the file to be uploaded for each part.

To upload the parts

1. When uploading a part, in addition to the upload ID, you must specify a part number by using the `--part-number` argument. You can choose any part number between 1 and 10,000. A part number uniquely identifies a part and its position in the object you are uploading. The part number that you choose must be in a consecutive sequence (for example, it can be 1, 2, or 3). If you upload a new part using the same part number as a previously uploaded part, the previously uploaded part is overwritten.
2. Use the `upload-part` command to upload each part of your multipart upload. The `--upload-id` is the same as it was in the output created by the `create-multipart-upload` command in Step 3. To upload the first part of your data, use the following command:

```
aws s3api upload-part --bucket amzn-s3-demo-bucket1 --key
 'census_data_file' --part-number 1 --body census-part00 --upload-id
 "cNV6KCSNANFZapz1LUGPC5XwUVi1n6yUoIeSP138sNOKPeMhpKQRrbT9k0ePmgoOTCj9K83T4e2Gb5hQvNoNpCKqy
 --checksum-algorithm SHA256
```

Upon completion of each `upload-part` command, you should see output like the following example:

```
{
    "ServerSideEncryption": "AES256",
    "ETag": "\"e611693805e812ef37f96c9937605e69\"",
    "ChecksumSHA256": "QLl8R4i4+SaJlrl8ZIcutc5TbZtwt2NwB8lTXkd3GH0="
}
```

3. For subsequent parts, increment the part number accordingly:

```
aws s3api upload-part --bucket amzn-s3-demo-bucket1 --key 'census_data_file' --
part-number <part-number> --body <file-path> --upload-id "<your-upload-id>" --
checksum-algorithm SHA256
```

For example, use the following command to upload the second part:

```
aws s3api upload-part --bucket amzn-s3-demo-bucket1 --key
 'census_data_file' --part-number 2 --body census-part01 --upload-id
```

```
"cNV6KCSNANFZapz1LUGPC5XwUVi1n6yUoIeSP138sNOKPeMhpKQRrbT9k0ePmgoOTCj9K83T4e2Gb5hQvNoNpCKqy
--checksum-algorithm SHA256
```

Amazon S3 returns an entity tag (ETag) and additional checksums for each uploaded part as a header in the response.

4. Continue using the `upload-part` command until you have uploaded all the parts of your object.

Step 5: List all the parts of your multipart upload

To complete the multipart upload, you will need a list of all the parts that have been uploaded for that specific multipart upload. The output from the `list-parts` command provides information such as bucket name, key, upload ID, part number, ETag, additional checksums, and more. It's helpful to save this output in a file so that you can use it for the next step when completing the multipart upload process. You can create a JSON output file called `parts.json` by using the following method.

To create a file that lists all of the parts

1. To generate a JSON file with the details of all the uploaded parts, use the following `list-parts` command. Replace ***amzn-s3-demo-bucket1*** with your actual bucket name and **<your-upload-id>** with the upload ID that you received in Step 3. For more information on the `list-parts` command, see list-parts in the *AWS Command Line Interface User Guide*.

```
aws s3api list-parts --bucket amzn-s3-demo-bucket1 --key 'census_data_file' --
upload-id <your-upload-id> --query '{Parts: Parts[*].{PartNumber: PartNumber, ETag:
 ETag, ChecksumSHA256: ChecksumSHA256}}' --output json > parts.json
```

A new file called `parts.json` is generated. The file contains the JSON formatted information for all of your uploaded parts. The `parts.json` file includes essential information for each part of your multipart upload, such as the part numbers and their corresponding ETag values, which are necessary for completing the multipart upload process.

2. Open `parts.json` by using any text editor or through the terminal. Here's the example output:

```
{
    "Parts": [
```

```
        {
            "PartNumber": 1,
            "ETag": "\"3c3097f89e2a2fece47ac54b243c9d97\"",
            "ChecksumSHA256": "fTPVHfyNHdv5VkR4S3EewdyioXECv7JBxN+d4FXYYTw="
        },
        {
            "PartNumber": 2,
            "ETag": "\"03c71cc160261b20ab74f6d2c476b450\"",
            "ChecksumSHA256": "VDWTa8enjOvULBAO3W2a6C+5/7ZnNjrnLApa1QVc3FE="
        },
        {
            "PartNumber": 3,
            "ETag": "\"81ae0937404429a97967dffa7eb4affb\"",
            "ChecksumSHA256": "cVVkXehUlzcwrBrXgPIM+EKQXPUvWist8mlUTCs4bg8="
        }
    ]
}
```

Step 6: Complete the multipart upload

After uploading all parts of your multipart upload and listing them, the final step is to complete the multipart upload. This step merges all the uploaded parts into a single object in your S3 bucket.

> **Note**
>
> You can calculate the object checksum before calling `complete-multipart-upload` by including `--checksum-sha256` in your request. If the checksums don't match, Amazon S3 fails the request. For more information, see complete-multipart-upload in the *AWS Command Line Interface User Guide*.

To complete the multipart upload

To finalize the multipart upload, use the `complete-multipart-upload` command. This command requires the `parts.json` file created in Step 5, your bucket name, and the upload ID. Replace **<*amzn-s3-demo-bucket1*>** with your bucket name and **<your-upload-id>** with the upload ID of `parts.json`.

```
aws s3api complete-multipart-upload --multipart-upload file://parts.json --bucket amzn-s3-demo-bucket1 --key 'census_data_file' --upload-id <your-upload-id>
```

Here's the example output:

```
{
    "ServerSideEncryption": "AES256",
    "Location": "https://amzn-s3-demo-bucket1.s3.us-east-2.amazonaws.com/census_data_file",
    "Bucket": "amzn-s3-demo-bucket1",
    "Key": "census_data_file",
    "ETag": "\"f453c6dccca969c457efdf9b1361e291-3\"",
    "ChecksumSHA256": "aI8EoktCdotjU8Bq46DrPCxQCGuGcPIhJ51noWs6hvk=-3"
}
```

Note

Don't delete the individual part files yet. You will need the individual parts so that you can perform checksums on them to verify the integrity of the merged-together object.

Step 7: Confirm that the object is uploaded to your bucket

After completing the multipart upload, you can verify that the object has been successfully uploaded to your S3 bucket. To list the objects in your bucket and confirm the presence of your newly uploaded file, use the `list-objects-v2` command

To list the uploaded object

To list the objects in your, use the `list-objects-v2` command bucket. Replace ***amzn-s3-demo-bucket1*** with your actual bucket name:

```
aws s3api list-objects-v2 --bucket amzn-s3-demo-bucket1
```

This command returns a list of objects in your bucket. Look for your uploaded file (for example, `census_data_file`) in the list of objects.

For more information, see the Examples section for the `list-objects-v2` command in the *AWS Command Line Interface User Guide*.

Step 8: Verify object integrity with an MD5 checksum

When you upload an object, you can specify a checksum algorithm for Amazon S3 to use. By default, Amazon S3 stores the MD5 digest of bytes as the object's ETag. For multipart uploads, the ETag is not the checksum for the entire object, but rather a composite of checksums for each individual part.

To verify object integrity by using an MD5 checksum

1. To retrieve the ETag of the uploaded object, perform a `head-object` request:

   ```
   aws s3api head-object --bucket amzn-s3-demo-bucket1 --key census_data_file
   ```

   Here's the example output:

   ```
   {
       "AcceptRanges": "bytes",
       "LastModified": "2024-07-26T19:04:13+00:00",
       "ContentLength": 16106127360,
       "ETag": "\"f453c6dccca969c457efdf9b1361e291-3\"",
       "ContentType": "binary/octet-stream",
       "ServerSideEncryption": "AES256",
       "Metadata": {}
   }
   ```

   This ETag has "-3" appended to the end. This indicates that the object was uploaded in three parts using multipart upload.

2. Next, calculate the MD5 checksum of each part using the `md5sum` command. Make sure that you provide the correct path to your part files:

   ```
   md5sum census-part*
   ```

   Here's the example output:

   ```
   e611693805e812ef37f96c9937605e69 census-part00
   63d2d5da159178785bfd6b6a5c635854 census-part01
   95b87c7db852451bb38b3b44a4e6d310 census-part02
   ```

3. For this step, manually combine the MD5 hashes into one string. Then, run the following command to convert the string to binary and calculate the MD5 checksum of the binary value:

```
echo
 "e611693805e812ef37f96c9937605e6963d2d5da159178785bfd6b6a5c63585495b87c7db852451bb38b3b44a
 | xxd -r -p | md5sum
```

Here's the example output:

```
f453c6dccca969c457efdf9b1361e291 -
```

This hash value should match the hash value of the original ETag value in Step 1, which validates the integrity of the `census_data_file` object.

When you instruct Amazon S3 to use additional checksums, Amazon S3 calculates the checksum value for each part and stores the values. If you want to retrieve the checksum values for individual parts of multipart uploads that are still in progress, you can use `list-parts`.

For more information about how checksums work with multipart upload objects, see Checking object integrity in Amazon S3.

Step 9: Verify object integrity with an additional checksum

In this step, this tutorial uses SHA-256 as an additional checksum to validate object integrity. If you've used a different additional checksum, use that checksum value instead.

To verify object integrity with SHA256

1. Run the following command in your terminal, including the `--checksum-mode enabled` argument, to display the `ChecksumSHA256` value of your object:

```
aws s3api head-object --bucket amzn-s3-demo-bucket1 --key census_data_file --
checksum-mode enabled
```

Here's the example output:

```
{
    "AcceptRanges": "bytes",
    "LastModified": "2024-07-26T19:04:13+00:00",
    "ContentLength": 16106127360,
    "ChecksumSHA256": "aI8EoktCdotjU8Bq46DrPCxQCGuGcPIhJ51noWs6hvk=-3",
    "ETag": "\"f453c6dccca969c457efdf9b1361e291-3\"",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {}
}
```

2. Use the following commands to decode the `ChecksumSHA256` values for the individual parts into base64 and save them into a binary file called `outfile`. These values can be found in your `parts.json` file. Replace the example base64 strings with your actual `ChecksumSHA256` values.

```
echo "QLl8R4i4+SaJlrl8ZIcutc5TbZtwt2NwB8lTXkd3GH0=" | base64 --decode >> outfile
echo "xCdgs1K5Bm4jWETYw/CmGYr+m6O2DcGfpckx5NVokvE=" | base64 --decode >> outfile
echo "f5wsfsa5bB+yXuwzqG1Bst91uYneqGD3CCidpb54mAo=" | base64 --decode >> outfile
```

3. Run the following command to calculate the SHA256 checksum of the `outfile`:

```
sha256sum outfile
```

Here's the example output:

```
688f04a24b42768b6353c06ae3a0eb3c2c50086b8670f221279d67a16b3a86f9 outfile
```

In the next step, take the hash value and convert it into a binary value. This binary value should match the `ChecksumSHA256` value from Step 1.

4. Convert the SHA256 checksum from Step 3 into binary, and then encode it to base64 to verify that it matches the `ChecksumSHA256` value from Step 1:

```
echo "688f04a24b42768b6353c06ae3a0eb3c2c50086b8670f221279d67a16b3a86f9" | xxd -r -p
 | base64
```

Here's the example output:

```
aI8EoktCdotjU8Bq46DrPCxQCGuGcPIhJ51noWs6hvk=
```

This output should confirm that the base64 output matches the `ChecksumSHA256` value from the `head-object` command output. If the output matches the checksum value, then the object is valid.

Important

- When you instruct Amazon S3 to use additional checksums, Amazon S3 calculates the checksum values for each part and stores these values.
- If you want to retrieve the checksum values for individual parts of multipart uploads that are still in progress, you can use the `list-parts` command.

Step 10: Clean up your resources

If you want to clean up the files created in this tutorial, use the following method. For instructions on deleting the files uploaded to your S3 bucket, see Deleting Amazon S3 objects.

Delete local files created in Step 1:

To remove the files that you created for your multipart upload, run the following command from your working directory:

```
rm census-data.bin census-part* outfile parts.json
```

Amazon S3 multipart upload limits

Multipart upload allows you to upload a single object as a set of parts. Each part is a contiguous portion of the object's data. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object. In general, when your object size reaches 100 MB, you should consider using multipart uploads instead of uploading the object in a single operation. For more information about multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3.

The following table provides multipart upload core specifications. These include maximum object size, maximum number of parts, maximum part size, and more. There is no minimum size limit on the last part of your multipart upload.

Item	Specification
Maximum object size	48.8 TiB
Maximum number of parts per upload	10,000
Part numbers	1 to 10,000 (inclusive)
Part size	5 MiB to 5 GiB. There is no minimum size limit on the last part of your multipart upload.
Maximum number of parts returned for a list parts request	1000
Maximum number of multipart uploads returned in a list multipart uploads request	1000

Add preconditions to S3 operations with conditional requests

You can use conditional requests to add preconditions to your S3 operations. To use conditional requests, you add an additional header to your Amazon S3 API operation. This header specifies a condition that, if not met, will result in the S3 operation failing.

Conditional reads are supported for `GET`, `HEAD`, and `COPY` requests. You can add preconditions to return or copy an object based on its Entity tag (ETag) or last modified date. This can limit an S3 operation to objects updated since a specified date. You can also limit an S3 operation to a specific ETag. This could ensure you only return or copy a specific object version. For more information about the object metadata, see Working with object metadata.

Conditional writes can ensure there is no existing object with the same key name in your bucket during `PUT` operations. This prevents overwriting of existing objects with identical key names. Similarly, you can use conditional writes to check if an object's ETag is unchanged before updating

the object. This prevents unintentional overwrites on an object without knowing the state of its content. You can use conditional writes for PutObject, CompleteMultipartUpload, or CopyObject requests. For more information about key names, see Naming Amazon S3 objects.

Conditional deletes evaluate if your object exists or is unchanged before deleting it. You can perform conditional deletes using the `DeleteObject` or `DeleteObjects` APIs in general purpose and directory buckets. For more information on conditional deletes, see How to perform conditional deletes. There is no additional charge for conditional reads, conditional writes or conditional deletes. You are only charged existing rates for the applicable requests, including for failed requests. For information about Amazon S3 features and pricing, see Amazon S3 pricing.

Topics

- How to retrieve or copy objects based on metadata with conditional reads
- How to prevent object overwrites with conditional writes
- How to perform conditional deletes

How to retrieve or copy objects based on metadata with conditional reads

With conditional reads, you can add an additional header to your read request in order to add preconditions to your S3 operation. If these preconditions are not met the read request will fail.

You can use conditional reads on `GET`, `HEAD`, or `COPY` requests to only return an object based on its metadata.

When you upload an object, Amazon S3 creates system controlled metadata that can only be modified by S3. Entity tags (ETags) and Last-Modified are examples of system controlled metadata. An object's ETag is a string representing a specific version of an object. Last-Modified date is metadata representing an object's creation date or the last modified date, whichever is the latest.

With conditional reads, you can return an object based on the object's ETag or Last-Modified date. You can specify an ETag value with your request and return the object only if the ETag value matches. This can ensure you only return or copy a specific version of an object. You can specify a Last-Modified value with your read request and return an object only if that object has been modified since a date you provide.

Supported APIs

The following S3 APIs support using conditional reads:

- GetObject
- HeadObject
- CopyObject

You can use the following headers to return an object dependent on the entity tag (ETag) or last modified date. For more information about object metadata such as ETags and Last-Modified, see the section called "System-defined object metadata".

GetObject

- `If-Match` — Return the object only if its ETag matches the one provided.
- `If-Modified-Since` — Return the object only if it has been modified since the time specified.
- `If-None-Match` — Return the object only if its ETag does not matches the one provided.
- `If-Unmodified-Since` — Return the object only if it has not been modified since the time specified.

For more information about these headers, errors returned, and the order S3 handles multiple conditional headers in a single request, see GetObject in the Amazon Simple Storage Service API Reference.

HeadObject

- `If-Match` — Return the object only if its ETag matches the one provided.
- `If-Modified-Since` — Return the object only if it has been modified since the time specified.
- `If-None-Match` — Return the object only if its ETag does not matches the one provided.
- `If-Unmodified-Since` — Return the object only if it has not been modified since the time specified.

For more information about these headers, errors returned, and the order S3 handles multiple conditional headers in a single request, see HeadObject in the Amazon Simple Storage Service API Reference.

CopyObject

- `x-amz-copy-source-if-match` — Copies the source object only if its ETag matches the one provided.
- `x-amz-copy-source-if-modified-since` — Copies the source object only if it has been modified since the time specified.
- `x-amz-copy-source-if-none-match` — Copies the source object only if its ETag does not matches the one provided.
- `x-amz-copy-source-if-unmodified-since` — Copies the source object only if it has not been modified since the time specified.
- `If-Match` — Copies the object only if its ETag matches the one provided. `If-Match` expects the ETag value as a string.
- `If-None-Match` — Copies the object only if its ETag does not match the one provided. `If-None-Match` expects the '*' (asterisk) character.

For more information about these headers, errors returned, and the order S3 handles multiple conditional headers in a single request, see CopyObject in the Amazon Simple Storage Service API Reference.

For information about using conditional headers to prevent overwrites during write operations, see How to prevent object overwrites with conditional writes.

How to prevent object overwrites with conditional writes

By using conditional writes, you can add an additional header to your `WRITE` requests to specify preconditions for your Amazon S3 operation. To conditionally write objects, add the HTTP `If-None-Match` or `If-Match` header.

The `If-None-Match` header prevents overwrites of existing data by validating that there's not an object with the same key name already in your bucket.

Alternatively, you can add the `If-Match` header to check an object's entity tag (ETag) before writing an object. With this header, Amazon S3 compares the provided ETag value with the ETag value of the object in S3. If the ETag values don't match, the operation fails.

Bucket owners can use bucket policies to enforce conditional writes for uploaded objects. For more information, see the section called “Enforce conditional writes”.

> **Note**
>
> To use conditional writes, you must use AWS Signature Version 4 to sign the request.

Topics

- How to prevent object overwrites based on key names
- How to prevent overwrites if the object has changed
- Conditional write behavior
- Conditional write scenarios
- Enforce conditional writes on Amazon S3 buckets

How to prevent object overwrites based on key names

You can use the HTTP `If-None-Match` conditional header to check whether an object already exists in the specified bucket based on its key name before creating it or copying it to the destination bucket.

Conditional writes with the HTTP `If-None-Match` header check for the existence of an object during the `WRITE` operation. If an identical key name is found in the bucket, the operation fails. Without the HTTP `If-None-Match` header, if you upload or copy an object with an identical key name in an unversioned or version-suspended bucket, the object is overwritten. For more information about using key names, see the section called “Naming objects”.

> **Note**
>
> The HTTP `If-None-Match` header only applies to the current version of an object in a version bucket.

To perform conditional writes with the HTTP `If-None-Match` header you must have the `s3:PutObject` permission. This enables the caller to check for the presence of objects in the bucket. The `If-None-Match` header expects the * (asterisk) value.

You can use the `If-None-Match` header with the following APIs:

- PutObject

- CompleteMultipartUpload
- CopyObject

Conditional Put Using the AWS CLI

The following `put-object` example command attempts to perform a conditional write for an object with the key name `dir-1/my_images.tar.bz2`.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key dir-1/my_images.tar.bz2 --body
 my_images.tar.bz2 --if-none-match "*"
```

For more information, see put-object in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

Conditional Copy Using the AWS CLI

The following `copy-object` example command attempts to copy an object to a destination bucket with a conditional write for an object with the key name `dir-1/my_images.tar.bz2`.

```
aws s3api copy-object --copy-source amzn-s3-demo-bucket/key --key dir-1/
my_images.tar.bz2 --bucket amzn-s3-demo-bucket2 --if-none-match "*"
```

For more information, see copy-object in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

Conditional multipart upload Using the AWS CLI

The following `complete-multipart-upload` example command attempts to complete a multipart upload with a conditional write for an object with the key name `dir-1/my_images.tar.bz2`. In this example, the file:// prefix is used to load the JSON structure from a file in the local folder named `mpustruct` which list of all the parts that have been uploaded for tnis specific multipart upload.

```
aws s3api complete-multipart-upload --multipart-upload file://mpustruct --bucket amzn-
s3-demo-bucket --key dir-1/my_images.tar.bz2 --upload-id upload-id  --if-none-match "*"
```

For more information, see complete-multipart-upload in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

How to prevent overwrites if the object has changed

An object's ETag is a string that's unique to the object and reflects a change to the object's content. You can use the `If-Match` header to compare the ETag value of an object in an Amazon S3 bucket with one that you provide during the `WRITE` operation. If the ETag values don't match, the operation fails. For more information about ETags, see the section called "Using Content-MD5 and the ETag to verify uploaded objects".

To perform conditional writes with an HTTP `If-Match` header you must have the `s3:PutObject` and `s3:GetObject` permissions. This enables the caller to check the ETag and verify the state of the objects in the bucket. The `If-Match` header expects the ETag value as a string.

You can use the `If-Match` header with the following APIs:

- PutObject
- CompleteMultipartUpload
- CopyObject

Conditional Put Using the AWS CLI

The following `put-object` example command attempts to perform a conditional write with the provided ETag value `6805f2cfc46c0f04559748bb039d69ae`.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key dir-1/my_images.tar.bz2 --body
 my_images.tar.bz2 --if-match "6805f2cfc46c0f04559748bb039d69ae"
```

For more information, see put-object in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

Conditional Copy Using the AWS CLI

The following `copy-object` example command attempts to perform a conditional write with the provided ETag value `6805f2cfc46c0f04559748bb039d69ae`.

```
aws s3api copy-object --copy-source amzn-s3-demo-bucket/key --key
 dir-1/my_images.tar.bz2 --bucket amzn-s3-demo-bucket2 --if-match
 "6805f2cfc46c0f04559748bb039d69ae"
```

For more information, see copy-object in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

Conditional multipart upload Using the AWS CLI

The following `complete-multipart-upload` example command attempts to complete a multipart upload with a conditional write using the provided ETag value `6805f2cfc46c0f04559748bb039d69ae`. In this example, the file:// prefix is used to load the JSON structure from a file in the local folder named `mpustruct` which list of all the parts that have been uploaded for tnis specific multipart upload.

```
aws s3api complete-multipart-upload --multipart-upload file://mpustruct --bucket amzn-
s3-demo-bucket --key dir-1/my_images.tar.bz2 --upload-id upload-id --if-match
 "6805f2cfc46c0f04559748bb039d69ae"
```

For more information, see complete-multipart-upload in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

Conditional write behavior

Conditional writes or copies with `If-None-Match` header

Conditional writes with the `If-None-Match` header evaluate against existing objects in a bucket. If there's no existing object with the same key name in the bucket, the write operation succeeds, resulting in a `200 OK` response. If there's an existing object, the write operation fails, resulting in a `412 Precondition Failed` response.

For buckets with versioning enabled, if there's no current object version with the same name, or if the current object version is a delete marker, the write operation succeeds. Otherwise, it results in a failed write operation with a `412 Precondition Failed` response.

If multiple conditional writes or copies occur for the same object name, the first write operation to finish succeeds. Amazon S3 then fails subsequent writes with a `412 Precondition Failed` response.

You can also receive a `409 Conflict` response in the case of concurrent requests if a delete request to an object succeeds before a conditional write operation on that object completes. When using conditional writes with `PutObject`, uploads may be retried after receiving a `409 Conflict` error. When using `CompleteMultipartUpload`, the entire multipart upload must be re-initiated with `CreateMultipartUpload` to upload the object again after receiving a `409 Conflict` error.

Conditional writes or copies with `If-Match` header

The `If-Match` header evaluates against existing objects in a bucket. If there's an existing object with the same key name and matching ETag, the write operation succeeds, resulting in a `200 OK` response. If the ETag doesn't match, the write operation fails with a `412 Precondition Failed` response.

You can also receive a `409 Conflict` response in the case of concurrent requests.

You will receive a `404 Not Found` response if a concurrent delete request to an object succeeds before a conditional write operation on that object completes, as the object key no longer exists. You should reupload the object when you receive a `404 Not Found` response.

If there's no current object version with the same name, or if the current object version is a delete marker, the operation fails with a `404 Not Found` error.

Conditional write scenarios

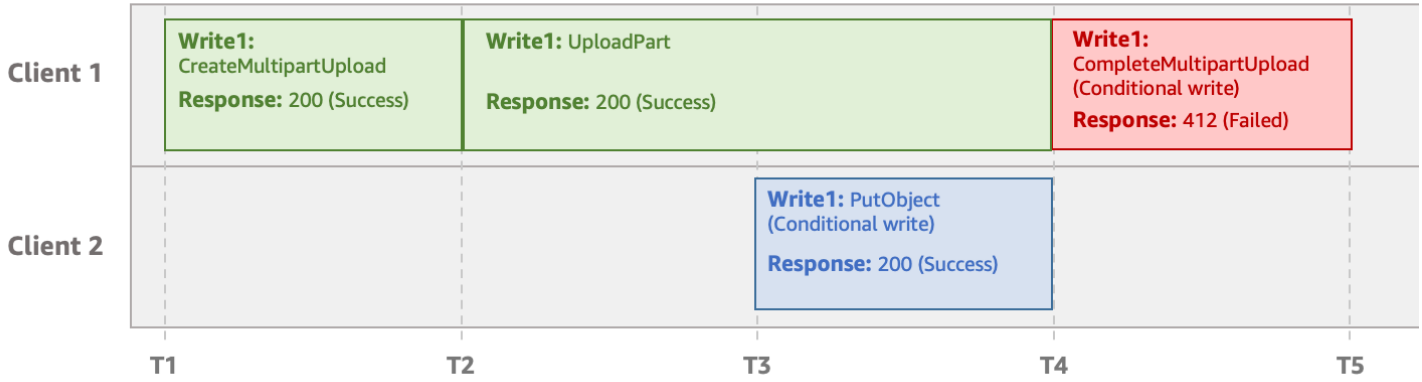
Consider the following scenarios where two clients are running operations on the same bucket.

Conditional writes during multipart uploads

Conditional writes do not consider any in-progress multipart uploads requests since those are not yet fully written objects. Consider the following example where Client 1 is uploading an object using multipart upload. During the multipart upload, Client 2 is able to successfully write the same object with the conditional write operation. Subsequently, when Client 1 tries to complete the multipart upload using a conditional write the upload fails.

> **Note**
>
> This scenario will result in a `412 Precondition Failed` response for both `If-None-Match` and `If-Match` headers.

The following example shows two clients writing items with the same key name. One with UploadPart for MPU and one with PutObject and a conditional write. The CompleteMultipartUpload operation, which starts after, fails.

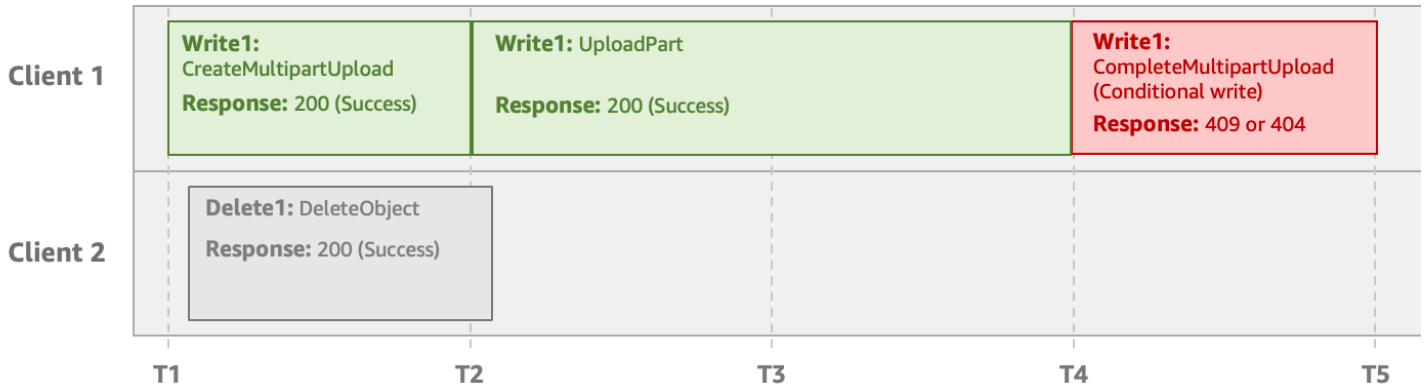


Concurrent deletes during multipart uploads

If a delete request succeeds before a conditional write request can complete, Amazon S3 returns a `409 Conflict` or `404 Not Found` response for the write operation. This is because the delete request that was initiated earlier takes precedence over the conditional write operation. In such cases, you must initiate a new multipart upload.

> **Note**
>
> This scenario will result in a `409 Conflict` response for an `If-None-Match` header and a `404 Not Found` response for an `If-Match` header.

The following example shows two clients, one using multipart upload and one sending a delete request after the MPU has started. The delete request finishes before the conditional write starts.



Note

To minimize your storage costs, we recommend that you configure a lifecycle rule to delete incomplete multipart uploads after a specified number of days by using the `AbortIncompleteMultipartUpload` action. For more information about creating a lifecycle rule to delete incomplete multipart uploads, see Configuring a bucket lifecycle configuration to delete incomplete multipart uploads.

Enforce conditional writes on Amazon S3 buckets

By using Amazon S3 bucket policies, you can enforce conditional writes for object uploads in your general purpose buckets.

A bucket policy is a resource-based policy that you can use to grant access permissions to your Amazon S3 bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. For more information about bucket policies, see the section called "Bucket policies".

You can use the condition keys `s3:if-match` or `s3:if-none-match` as the optional `Condition` element or `Condition` block to specify when a policy is in effect. For multipart uploads you must specify the `s3:ObjectCreationOperation` condition key to exempt the `CreateMultipartUpload`, `UploadPart`, and `UploadPartCopy` operations, as these APIs don't accept conditional headers. For more information about using conditions in bucket policies, see the section called "Condition key examples".

> **Note**
>
> If you use a bucket policy to enforce conditional writes, you can't perform copy operations to the bucket or prefix specified in your bucket policy. `CopyObject` requests without an `If-None-Match` or `If-Match` HTTP header fail with a `403 Access Denied` error. `CopyObject` requests made with those HTTP headers fail with a `501 Not Implemented` response.

The following examples show how to use conditions in a bucket policy to force clients to use the `If-None-Match` or `If-Match` HTTP header.

Topics

- Example 1: Only allow object uploads using PutObject and CompleteMultipartUpload requests that include the if-none-match header
- Example 2: Only allow object uploads using PutObject and CompleteMultipartUpload requests that include the if-match header
- Example 3: Only allow object upload requests that includes the if-none-match or if-match header

Example 1: Only allow object uploads using `PutObject` and `CompleteMultipartUpload` requests that include the `if-none-match` header

This policy allows account 111122223333, user Alice, to write to the *`amzn-s3-demo-bucket1`* bucket if the request includes the `if-none-match` header, ensuring that the object key doesn't already exist in the bucket. All `PutObject` and `CompleteMultipartUpload` requests to the specified bucket must include the `if-none-match` header to succeed. Using this header, customers can write to this bucket only if the object key does not exist in the bucket.

> **Note**
>
> This policy also sets the `s3:ObjectCreationOperation` condition key which allows for multipart uploads using the `CreateMultipartUpload`, `UploadPart`, and `UploadPartCopy` APIs.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowConditionalPut",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Null": {
                    "s3:if-none-match": "false"
                }
            }
        },
        {
            "Sid": "AllowConditionalPutwithMPUs",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Bool": {
                    "s3:ObjectCreationOperation": "false"
                }
            }
        }
    ]
}
```

Example 2: Only allow object uploads using `PutObject` and `CompleteMultipartUpload` requests that include the `if-match` header

This policy allows account 111122223333, user Alice to write to *amzn-s3-demo-bucket1* only if the request includes the `if-match` header. This header compares the ETag value of an object in S3

with one you provide during the `WRITE` operation. If the ETag values do not match, the operation will fail. All `PutObject` and `CompleteMultipartUpload` requests to the specified bucket must include the `if-match` header to succeed.

> **Note**
>
> This policy also sets the `s3:ObjectCreationOperation` condition key which allows for multipart uploads using the `CreateMultipartUpload`, `UploadPart`, and `UploadPartCopy` APIs.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObject",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        },
        {
            "Sid": "BlockNonConditionalObjectCreation",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Null": {
                    "s3:if-match": "true"
                },
                "Bool": {
                    "s3:ObjectCreationOperation": "true"
                }
            }
        },
        {
            "Sid": "AllowGetObjectBecauseConditionalPutIfMatchETag",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        }
    ]
}
```

Example 3: Only allow object upload requests that includes the `if-none-match` or `if-match` header

This policy allows account 111122223333, user Alice to write to *amzn-s3-demo-bucket1* if the requests include the `if-none-match` or `if-match` header. This allows Alice to upload an object if the key name does not exist in the bucket, or if the key name does exist, Alice can overwrite the object if the object ETag matches the ETag provided in the PUT request.

> **Note**
>
> This policy also sets the `s3:ObjectCreationOperation` condition key which allows for multipart uploads using the `CreateMultipartUpload`, `UploadPart`, and `UploadPartCopy` APIs.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": " AllowConditionalPutifAbsent",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Null": {
                    "s3:if-none-match": "false"
```

```
                }
            }
        },
        {
            "Sid": "AllowConditionalPutIfMatchEtag",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Null": {
                    "s3:if-match": "false"
                }
            }
        },
        {
            "Sid": "AllowConditionalObjectCreation",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "Bool": {
                    "s3:ObjectCreationOperation": "false"
                }
            }
        },
        {
            "Sid": " AllowGetObjectBecauseConditionalPutIfMatchETag",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        }
    ]
}
```

How to perform conditional deletes

You can use conditional deletes to evaluate if your object exists or is unchanged before deleting it. You can perform conditional deletes using the `DeleteObject` or `DeleteObjects` API operations in S3 general purpose and directory buckets. To get started, when making a conditional delete request, you can use the `HTTP If-Match` header with the precondition value `*` to check if object exists or the `If-Match` header with your provided `ETag` to check if the object has been modified.

You can enforce conditional deletes at a general purpose bucket level using S3 bucket or Identity and Access Management (IAM) policies. For more information, see Enforce conditional deletes on Amazon S3 buckets.

> **Note**
>
> Conditional delete evaluations only apply to the current version of the object.

Topics

- How to check if your object has been modified before deleting it
- How to check if your object exists before deleting it
- Enforce conditional deletes on Amazon S3 buckets

How to check if your object has been modified before deleting it

With conditional deletes, you can protect your application from accidental deletions of objects. You can use the `HTTP If-Match` header with the `ETag` value to check if an object has been modified. If the `ETag` value of an object in an S3 bucket doesn't match with the `ETag` that you provide during the delete operation, the operation fails. For conditionally deleting multiple objects using the `DeleteObjects` operation, you must provide the `ETag` value in the `ETag` element of the object in the XML request body. For more information, see the section called "Using Content-MD5 and the ETag to verify uploaded objects".

> **Note**
>
> To perform conditional deletes with the `If-Match` header with the `ETag` value, you must have the `s3:DeleteObject` and `s3:GetObject` permissions.

The `If-Match` header with the `ETag` value evaluates against existing objects in a bucket. If there's an existing object with the same key name and matching `ETag`, the `DeleteObject` requests succeeds, and returns a `204 No content` response. If the `ETag` doesn't match, the delete operation fails with a `412 Precondition Failed` response. To conditionally delete multiple objects using the `DeleteObjects` operation, you can provide the `ETag` value in the `ETag` element of the object in the XML request body. If the request succeeds, the `DeleteObjects` operation responds with a `200 OK` and provides the status of each object in the response body. If the precondition succeeds, the response for that object will be captured in the `<Deleted>` element of the response body. If the precondition fails then the response for that object will be captured in the `<Error>` element of the response body.

You can also receive a `409 Conflict` error response in the case of concurrent requests if a `DELETE` or `PUT` request to an object succeeds before a conditional delete operation on that object completes. You will receive a `404 Not Found` response if a concurrent delete request to an object succeeds before a conditional write operation on that object completes, as the object key no longer exists.

You can use the `If-Match` header with the `ETag` value for the following APIs:

- DeleteObject
- DeleteObjects

Using the AWS CLI

The following `delete-object` example command attempts to perform a conditional delete with the provided ETag value `6805f2cfc46c0f04559748bb039d69al`.

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key dir-1/my_images.tar.bz2 --
if-match "6805f2cfc46c0f04559748bb039d69al"
```

For more information, see delete-object in the *AWS CLI Command Reference*.

The following `delete-objects` example command attempts to perform a conditional delete with the provided ETag value `6805f2cfc46c0f04559748bb039d69al`.

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete '{"Objects":
[{"Key":"my_images.tar.bz2", "ETag": "6805f2cfc46c0f04559748bb039d69al"}]}'
```

For more information, see delete-objects in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

How to check if your object exists before deleting it

You can use the `If-Match` header with the `*` value to check if the object exists before attempting to delete it. The `*` value signifies that the operation should only proceed if the object exists, regardless of whether it has been modified or not.

Delete markers are special objects in versioned S3 general purpose buckets that indicate that an object has been deleted. They are placeholders that make the object appear deleted while preserving the previous versions. Therefore, when you use `If-Match:*` with a `DeleteObject` API, the operation will only succeed with a `204 No Content` if the object exists. If the latest version of the object is a delete marker, the object doesn't exist and the `DeleteObject` API will fail and return a `412 Precondition Failed` response. For more information about delete markers, see the section called "Working with delete markers".

To conditionally delete multiple objects using the `DeleteObjects` operation, you can provide the `*` in the `ETag` element of the object in the XML request body. If the precondition succeeds, the `DeleteObjects` operation responds with a `200 OK` and provides the status of each object in the response body. If the precondition succeeds, the response for that object will be captured in the `<Deleted>` element of the response body. If the precondition fails then the response for that object will be captured in the `<Error>` element of the response body. If the object doesn't exist when evaluating either of the preconditions, S3 rejects the request and returns a `Not Found` error response.

> **Note**
>
> To perform conditional deletes with `If-Match:*`, you must have `s3:DeleteObject` permissions.

You can use the `If-Match` header with the `*` value for the following APIs:

- DeleteObject
- DeleteObjects

Using the AWS CLI

The following `delete-object` example command attempts to perform a conditional delete for an object with the key name *`my_images.tar.bz2`* that has a value of `*` which represents any ETag.

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key dir-1/my_images.tar.bz2 --
if-match "*"
```

For more information, see delete-object in the *AWS CLI Command Reference*.

The following `delete-objects` example command attempts to perform a conditional delete for an object with the key name *`my_images.tar.bz2`* that has a value of `*` which represents any ETag.

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete '{"Objects":
[{"Key":"my_images.tar.bz2", "ETag": "*"}]}'
```

For more information, see delete-objects in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

Enforce conditional deletes on Amazon S3 buckets

By using Amazon S3 bucket policies, you can enforce `If-Matchheader` with conditional deletes for objects in general purpose buckets. If the `If-Match` header doesn't exist, the request will be denied with an `403 Access Denied`. A bucket policy is a resource-based policy that you can use to grant access permissions to your bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. For more information about bucket policies, see the section called "Bucket policies".

The following examples show how to use conditions in a bucket policy to force clients to use the `If-Match` HTTP header.

Topics

- Example 1: Only allow conditional deletes using the If-Match header with the ETag value
- Example 2: Only allow conditional deletes using the If-Match header with the * value

Example 1: Only allow conditional deletes using the `If-Match` header with the `ETag` value

You can use this bucket policy to only allow conditional deletes using `DeleteObject` and `DeleteObjects` requests that include the `If-Match` header with the `ETag` value. The `Null` condition ensures the `If-Match` header is present, and the `s3:GetObject` permission is granted because conditional deletes with a specific ETag value require both `s3:DeleteObject` and `s3:GetObject` permissions. All non-conditional deletes would be denied and conditional deletes would pass.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowConditionalDeletes",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:DeleteObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Null": {
                    "s3:if-match": "false"
                }
            }
        },
         {
            "Sid": "AllowGetObjectBecauseConditionalDeleteIfMatchETag",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

Example 2: Only allow conditional deletes using the `If-Match` header with the `*` value

You can use this bucket policy to only allow conditional deletes using `DeleteObject` and `DeleteObjects` requests that include the `If-Match` header with the `*` value. The `Null` condition

ensures the `If-Match` header is present. Because `s3:GetObject` is not granted, conditional deletes with a specific ETag value will fail – only `If-Match: *` (which checks object existence and requires only `s3:DeleteObject` permission) will succeed. All non-conditional deletes would be denied, and only `If-Match: *` conditional deletes would succeed.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowConditionalDeletes",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Alice"
            },
            "Action": "s3:DeleteObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Null": {
                    "s3:if-match": "false"
                }
            }
        }
    ]
}
```

Copying, moving, and renaming objects

The `CopyObject` operation creates a copy of an object that's already stored in Amazon S3.

You can create a copy of an object up to 5 GB in a single atomic operation. However, to copy an object that's larger than 5 GB, you must use a multipart upload using the AWS CLI or AWS SDKs. For more information, see Copying an object using multipart upload.

> **Note**
>
> To maintain the performance benefit of an object you uploaded using multipart upload, you must copy the object using multipart upload using the AWS CLI or AWS SDK instead of the S3 console. For more information, see Copying an object using multipart upload.

Using the `CopyObject` operation, you can:

- Create additional copies of objects.
- Rename objects by copying them and deleting the original ones.
- Copy or move objects from one bucket to another, including across AWS Regions (for example, from `us-west-1` to `eu-west-2`). When you move an object, Amazon S3 copies the object to the specified destination and then deletes the source object.

  > **Note**
  >
  > Copying or moving objects across AWS Regions incurs bandwidth charges. For more information, see Amazon S3 Pricing.

- Change object metadata. Each Amazon S3 object has metadata. This metadata is a set of name-value pairs. You can set object metadata at the time you upload an object. After you upload the object, you can't modify the object metadata. The only way to modify object metadata is to make a copy of the object and set the metadata. To do so, in the copy operation, set the same object as the source and target.

  Some object metadata is system metadata and other is user-defined. You can control some of the system metadata. For example, you can control the storage class and the type of server-side encryption to use for the object. When you copy an object, user-controlled system metadata and user-defined metadata are also copied. Amazon S3 resets the system-controlled metadata. For example, when you copy an object, Amazon S3 resets the creation date of the copied object. You don't need to set any of these system-controlled metadata values in your copy request.

  When copying an object, you might decide to update some of the metadata values. For example, if your source object is configured to use S3 Standard storage, you might choose to use S3 Intelligent-Tiering for the object copy. You might also decide to alter some of the user-defined metadata values present on the source object. If you choose to update any of the object's user-configurable metadata (system or user-defined) during the copy, then you must explicitly specify all of the user-configurable metadata present on the source object in your request, even if you are changing only one of the metadata values.

  > **Note**
  >
  > When copying an object by using the Amazon S3 console, you might receive the error message "Copied metadata can't be verified." The console uses headers to retrieve and

set metadata for your object. If your network or browser configuration modifies your network requests, this behavior might cause unintended metadata (such as modified `Cache-Control` headers) to be written to your copied object. Amazon S3 can't verify this unintended metadata.
To address this issue, check your network and browser configuration to make sure it doesn't modify headers, such as `Cache-Control`. For more information, see The Shared Responsibility Model.

For more information about object metadata, see Working with object metadata.

Copying archived and restored objects

If the source object is archived in S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, you must first restore a temporary copy before you can copy the object to another bucket. For information about archiving objects, see Working with archived objects.

The **Copy** operation in the Amazon S3 console isn't supported for restored objects in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. To copy these restored objects, use the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

Copying encrypted objects

Amazon S3 automatically encrypts all new objects that are copied to an S3 bucket. If you don't specify encryption information in your copy request, the encryption setting of the target object is set to the default encryption configuration of the destination bucket. By default, all buckets have a base level of encryption configuration that uses server-side encryption with Amazon S3 managed keys (SSE-S3). If the destination bucket has a default encryption configuration that uses server-side encryption with an AWS Key Management Service (AWS KMS) key (SSE-KMS), or a customer-provided encryption key (SSE-C), Amazon S3 uses the corresponding KMS key, or a customer-provided key to encrypt the target object copy.

When copying an object, if you want to use a different type of encryption setting for the target object, you can request that Amazon S3 encrypt the target object with a KMS key, an Amazon S3 managed key, or a customer-provided key. If the encryption setting in your request is different from the default encryption configuration of the destination bucket, the encryption setting in your request takes precedence. If the source object for the copy is encrypted with SSE-C, you must

provide the necessary encryption information in your request so that Amazon S3 can decrypt the object for copying. For more information, see Protecting data with encryption.

Using checksums when copying objects

When copying objects, you can choose to use a different checksum algorithm for the object. Whether you choose to use the same algorithm or a new one, Amazon S3 calculates a new checksum value after the object is copied. Amazon S3 doesn't directly copy the value of the checksum. All copied objects without checksums and specified destination checksum algorithms automatically gain a `CRC-64NVME` checksum algorithm. For more information about how the checksum is calculated, see Uploading and copying objects using multipart upload in Amazon S3.

Copying multiple objects in a single request

To copy more than one Amazon S3 object with a single request, you can also use S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on. S3 Batch Operations calls the respective API operation to perform the specified operation. A single Batch Operations job can perform the specified operation on billions of objects containing exabytes of data.

The S3 Batch Operations feature tracks progress, sends notifications, and stores a detailed completion report of all actions, providing a fully managed, auditable, serverless experience. You can use S3 Batch Operations through the Amazon S3 console, AWS CLI, AWS SDKs, or REST API. For more information, see the section called “Batch Operations basics”.

Copying objects to directory buckets

For information about copying an object to a directory bucket, see Copying objects from or to a directory bucket. For information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Copying annotations

When you copy an object using the `CopyObject` API, Amazon S3 copies the object's annotations to the destination by default. You can exclude annotations from the copy by specifying the `x-amz-annotation-directive: EXCLUDE` header in the copy request.

When you copy objects using multipart upload (for example, with the AWS CLI `aws s3 cp` command for objects larger than approximately 8 MB), annotations are not copied by default. To include annotations when copying objects with multipart upload, specify `--copy-props all`.

For more information about annotations, see Annotating your objects.

To copy an object

To copy an object, use the following methods.

Using the S3 console

Note

The restrictions and limitations when you copy an object with the console are as follows:

- You can copy an object if your object is less than 5 GB. If your object is greater than 5 GB, you must use the AWS CLI or AWS SDKs to copy an object.
- For a list of additional permissions required to copy objects, see the section called "Required permissions for S3 API operations". For example policies that grant this permission, see the section called "Identity-based policy examples".
- The Copy action applies to all objects within the specified folders (prefixes). Objects added to these folders while the action is in progress might be affected.
- Cross-Region copying of objects encrypted with SSE-KMS is not supported by the Amazon S3 console. To copy objects encrypted with SSE-KMS across Regions, use the AWS CLI, AWS SDK, or the Amazon S3 REST API.
- Objects encrypted with customer-provided encryption keys (SSE-C) cannot be copied by using the S3 console. To copy objects encrypted with SSE-C, use the AWS CLI, AWS SDK, or the Amazon S3 REST API.
- Copied objects will not retain the Object Lock settings from the original objects.
- If the bucket you are copying objects from uses the bucket owner enforced setting for S3 Object Ownership, object ACLs will not be copied to the specified destination.
- If you want to copy objects to a bucket that uses the bucket owner enforced setting for S3 Object Ownership, make sure that the source bucket also uses the bucket owner enforced setting, or remove any object ACL grants to other AWS accounts and groups.

To copy an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **General purpose buckets** or **Directory buckets**.
3. In the list of buckets, choose the name of the bucket that contains the objects that you want to copy.
4. Select the check box to the left of the names of the objects that you want to copy.
5. On the **Actions** menu, choose **Copy** from the list of options that appears.
6. Select the destination type and destination account. To specify the destination path, choose **Browse S3**, navigate to the destination, and select the check box to the left of the destination. Choose **Choose destination**.

   Alternatively, enter the destination path.
7. If you do *not* have bucket versioning enabled, you will see a warning recommending you enable Bucket Versioning to help protect against unintentionally overwriting or deleting objects. If you want to keep all versions of objects in this bucket, select **Enable Bucket Versioning**. You can also view the default encryption and S3 Object Lock properties in **Destination details**.
8. Under **Additional copy settings**, choose whether you want to **Copy source settings**, **Don't specify settings**, or **Specify settings**. **Copy source settings** is the default option. If you only want to copy the object without the source settings attributes, choose **Don't specify settings**. Choose **Specify settings** to specify settings for storage class, ACLs, object tags, annotations, metadata, server-side encryption, and additional checksums.
9. Choose **Copy**. Amazon S3 copies your objects to the destination.

Using the AWS SDKs

The examples in this section show how to copy objects up to 5 GB in a single operation. To copy objects larger than 5 GB, you must use a multipart upload. For more information, see Copying an object using multipart upload.

Java

For examples of how to copy objects with the AWS SDK for Java, see Copy an object from one bucket to another in the *Amazon S3 API Reference*.

.NET

The following C# example uses the high-level SDK for .NET to copy objects that are as large as 5 GB in a single operation. For objects that are larger than 5 GB, use the multipart upload copy example described in Copying an object using multipart upload.

This example makes a copy of an object that is a maximum of 5 GB. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CopyObjectTest
    {
        private const string sourceBucket = "*** provide the name of the bucket with
 source object ***";
        private const string destinationBucket = "*** provide the name of the bucket
 to copy the object to ***";
        private const string objectKey = "*** provide the name of object to copy
 ***";
        private const string destObjectKey = "*** provide the destination object key
 name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            Console.WriteLine("Copying an object");
            CopyingObjectAsync().Wait();
        }

        private static async Task CopyingObjectAsync()
        {
            try
            {
                CopyObjectRequest request = new CopyObjectRequest
                {
                    SourceBucket = sourceBucket,
                    SourceKey = objectKey,
                    DestinationBucket = destinationBucket,
```

```
                    DestinationKey = destObjectKey
                };
                CopyObjectResponse response = await
 s3Client.CopyObjectAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

This topic guides you through using classes from version 3 of the AWS SDK for PHP to copy a single object and multiple objects within Amazon S3, from one bucket to another or within the same bucket.

For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

The following PHP example illustrates the use of the `copyObject()` method to copy a single object within Amazon S3. It also demonstrates how to make multiple copies of an object by using a batch of calls to `CopyObject` with the `getcommand()` method.

Copying objects

1	Create an instance of an Amazon S3 client by using the `Aws\S3\S3Client` class constructor.
2	To make multiple copies of an object, you run a batch of calls to the Amazon S3 client getCommand() method, which is inherited from the Aws\CommandInterface class. You provide the `CopyObject` command as the first argument and an array containing the source bucket, source key name, target bucket, and target key name as the second argument.

```
 require 'vendor/autoload.php';

use Aws\CommandPool;
use Aws\Exception\AwsException;
use Aws\ResultInterface;
use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';
$targetBucket = '*** Your Target Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

// Copy an object.
$s3->copyObject([
    'Bucket' => $targetBucket,
    'Key' => "$sourceKeyname-copy",
    'CopySource' => "$sourceBucket/$sourceKeyname",
]);

// Perform a batch of CopyObject operations.
$batch = array();
for ($i = 1; $i <= 3; $i++) {
    $batch[] = $s3->getCommand('CopyObject', [
        'Bucket' => $targetBucket,
        'Key' => "{targetKeyname}-$i",
        'CopySource' => "$sourceBucket/$sourceKeyname",
    ]);
}
try {
    $results = CommandPool::batch($s3, $batch);
    foreach ($results as $result) {
        if ($result instanceof ResultInterface) {
            // Result handling here
        }
        if ($result instanceof AwsException) {
            // AwsException handling here
        }
    }
} catch (Exception $e) {
```

```
    // General error handling here
}
```

Python

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource in
 Boto3
                          that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key
```

```
    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
 key.
                            This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
 self.object.key}
            )
            dest_object.wait_until_exists()
            logger.info(
                "Copied object from %s:%s to %s:%s.",
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
        except ClientError:
            logger.exception(
                "Couldn't copy object from %s/%s to %s/%s.",
```

```
                self.object.bucket_name,
                self.object.key,
                dest_object.bucket_name,
                dest_object.key,
            )
            raise
```

Ruby

The following tasks guide you through using the Ruby classes to copy an object in Amazon S3 from one bucket to another or within the same bucket.

Copying objects

1	Use the Amazon S3 modularized gem for version 3 of the AWS SDK for Ruby, require `aws-sdk-s3` , and provide your AWS credentials. For more information about how to provide your credentials, see Making requests using AWS account or IAM user credentials in the *Amazon S3 API Reference*.
2	Provide the request information, such as the source bucket name, source key name, destination bucket name, and destination key.

The following Ruby code example demonstrates the preceding tasks by using the `#copy_object` method to copy an object from one bucket to another.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                       copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
```

```
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Using the REST API

This example describes how to copy an object by using the Amazon S3 REST API. For more information about the REST API, see CopyObject.

This example copies the `flotsam` object from the *`amzn-s3-demo-bucket1`* bucket to the `jetsam` object of the *`amzn-s3-demo-bucket2`* bucket, preserving its metadata.

```
PUT /jetsam HTTP/1.1
Host: amzn-s3-demo-bucket2.s3.amazonaws.com
x-amz-copy-source: /amzn-s3-demo-bucket1/flotsam
Authorization: AWS AKIAIOSFODNN7EXAMPLE:ENoSbxYByFA0UGLZUqJN5EUnLDg=
Date: Wed, 20 Feb 2008 22:12:21 +0000
```

The signature was generated from the following information.

```
PUT\r\n
\r\n
\r\n
Wed, 20 Feb 2008 22:12:21 +0000\r\n

x-amz-copy-source:/amzn-s3-demo-bucket1/flotsam\r\n
/amzn-s3-demo-bucket2/jetsam
```

Amazon S3 returns the following response that specifies the ETag of the object and when it was last modified.

```
HTTP/1.1 200 OK
x-amz-id-2: Vyaxt7qEbzv34BnSu5hctyyNSlHTYZFMWK4FtzO+iX8JQNyaLdTshL0KxatbaOZt
x-amz-request-id: 6B13C3C5B34AF333
Date: Wed, 20 Feb 2008 22:13:01 +0000

Content-Type: application/xml
Transfer-Encoding: chunked
Connection: close
Server: AmazonS3
<?xml version="1.0" encoding="UTF-8"?>

<CopyObjectResult>
   <LastModified>2008-02-20T22:13:01</LastModified>
   <ETag>"7e9c608af58950deeb370c98608ed097"</ETag>
</CopyObjectResult>
```

Using the AWS CLI

You can also use the AWS Command Line Interface (AWS CLI) to copy an S3 object. For more information, see copy-object in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

To move an object

To move an object, use the following methods.

Using the S3 console

Note

- You can move an object if your object is less than 5 GB. If your object is greater than 5 GB, you must use the AWS CLI or AWS SDKs to move an object.
- For a list of additional permissions required to move objects, see the section called "Required permissions for S3 API operations". For example policies that grant this permission, see the section called "Identity-based policy examples".
- Objects encrypted with customer-provided encryption keys (SSE-C) can't be moved by using the Amazon S3 console. To move objects encrypted with SSE-C, use the AWS CLI, AWS SDKs, or the Amazon S3 REST API.
- When moving folders, wait for the **Move** operation to finish before making additional changes in the folders.
- You can't use S3 access point aliases as the source or destination for **Move** operations in the Amazon S3 console.

To move an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**. Navigate to the Amazon S3 bucket or folder that contains the objects that you want to move.
3. Select the check box for the objects that you want to move.
4. On the **Actions** menu, choose **Move**.
5. To specify the destination path, choose **Browse S3**, navigate to the destination, and select the destination check box. Choose **Choose destination**.

   Alternatively, enter the destination path.
6. If you do *not* have bucket versioning enabled, you will see a warning recommending you enable Bucket Versioning to help protect against unintentionally overwriting or deleting

objects. If you want to keep all versions of objects in this bucket, select **Enable Bucket Versioning**. You can also view the default encryption and Object Lock properties in **Destination details**.

7. Under **Additional copy settings**, choose whether you want to **Copy source settings**, **Don't specify settings**, or **Specify settings**. **Copy source settings** is the default option. If you only want to copy the object without the source settings attributes, choose **Don't specify settings**. Choose **Specify settings** to specify settings for storage class, ACLs, object tags, metadata, server-side encryption, and additional checksums.
8. Choose **Move**. Amazon S3 moves your objects to the destination.

Using the AWS CLI

You can also use the AWS Command Line Interface (AWS CLI) to move an S3 object. For more information, see mv in the *AWS CLI Command Reference*.

For information about the AWS CLI, see What is the AWS Command Line Interface? in the *AWS Command Line Interface User Guide*.

To rename an object

To rename an object, use the following procedure.

Note

- You can rename an object if your object is less than 5 GB. To rename objects greater than 5 GB, you must use the AWS CLI or AWS SDKs to copy your object with a new name and then delete the original object.
- For a list of additional permissions required to copy objects, see the section called "Required permissions for S3 API operations". For example policies that grant this permission, see the section called "Identity-based policy examples".
- Renaming an object creates a copy of the object with a new last-modified date, and then adds a delete marker to the original object.
- Bucket settings for default encryption are automatically applied to any specified object that's unencrypted.
- You can't use the Amazon S3 console to rename objects with customer-provided encryption keys (SSE-C). To rename objects encrypted with SSE-C, use the AWS CLI, AWS SDKs, or the Amazon S3 REST API to copy those objects with new names.

- If this bucket uses the bucket owner enforced setting for S3 Object Ownership, object access control lists (ACLs) won't be copied.

To rename an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Buckets**, and then choose the **General purpose buckets** tab. Navigate to the Amazon S3 bucket or folder that contains the object that you want to rename.
3. Select the check box for the object that you want to rename.
4. On the **Actions** menu, choose **Rename object**.
5. In the **New object name** box, enter the new name for the object.
6. Under **Additional copy settings**, choose whether you want to **Copy source settings**, **Don't specify settings**, or **Specify settings**. **Copy source settings** is the default option. If you only want to copy the object without the source settings attributes, choose **Don't specify settings**. Choose **Specify settings** to specify settings for storage class, ACLs, object tags, metadata, server-side encryption, and additional checksums.
7. Choose **Save changes**. Amazon S3 renames your object.

Downloading objects

This section explains how to download objects from an Amazon S3 bucket. With Amazon S3, you can store objects in one or more buckets, and each single object can be up to 50 TB in size. Any Amazon S3 object that is not archived is accessible in real time. Archived objects, however, must be restored before they can be downloaded. For information about downloading archived objects, see the section called “Downloading archived objects”.

You can download a single object by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API. To download an object up to 5 TB from S3 without writing any code or running any commands, use the S3 console. For more information, see the section called “Downloading an object”.

To download objects larger than 5 TB, use concurrent `GetObject` requests with either `Range` HTTP header to read specific byte ranges or `partNumber` to download specific part of an object.

Single GET requests are limited to 5 TB, and you will receive a `405 - Method Not Allowed` error for GET requests beyond 5 TB.

For large object downloads, use the S3 Transfer Manager in the Java v1/v2, Python, or AWS CLI SDKs. For the best performance, use the latest AWS Common Runtime (CRT) with these SDKs, which has been optimized for better resource utilization. CRT automatically scales the size of individual GETs to optimize throughput. You can improve overall transfer throughput by allocating more memory using memory limit parameters such as `maxNativeMemoryLimitInBytes` for Java SDK. You can opt out of this behavior by setting an explicit part size using request parameters such as `multipart_chunksize` for AWS CLI and `minimumPartSizeInBytes` for Java SDK in your download request.

To download multiple objects, use AWS CloudShell, the AWS CLI, or the AWS SDKs. For more information, see the section called “Downloading multiple objects”.

If you need to download part of an object, you use extra parameters with the AWS CLI or REST API to specify only the bytes that you want to download. For more information, see the section called “Downloading part of an object”.

If you need to download an object that you don't own, ask the object owner to generate a presigned URL that allows you to download the object. For more information, see the section called “Downloading an object from another AWS account”.

When you download objects outside of the AWS network, data-transfer fees apply. Data transfer within the AWS network is free within the same AWS Region, but you will be charged for any `GET` requests. For more information about data-transfer costs and data-retrieval charges, see Amazon S3 pricing.

Topics

- Downloading an object
- Downloading multiple objects
- Downloading part of an object
- Downloading an object from another AWS account
- Downloading archived objects
- Downloading objects based on metadata
- Troubleshooting downloading objects

Downloading an object

You can download an object by using the Amazon S3 console, AWS CLI, AWS SDKs, or REST API.

Using the S3 console

This section explains how to use the Amazon S3 console to download an object from an S3 bucket.

> **Note**
>
> - You can download only one object at a time.
> - If you use the Amazon S3 console to download an object whose key name ends with a period (`.`), the period is removed from the key name of the downloaded object. To retain the period at the end of the name of the downloaded object, you must use the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

To download an object from an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets** or **Directory buckets**.
3. In the buckets list, choose the name of the bucket that you want to download an object from.
4. You can download an object from an S3 bucket in any of the following ways:
   - Select the check box next to the object, and choose **Download**. If you want to download the object to a specific folder, on the **Actions** menu, choose **Download as**.
   - If you want to download a specific version of the object, turn on **Show versions** (located next to the search box). Select the check box next to the version of the object that you want, and choose **Download**. If you want to download the object to a specific folder, on the **Actions** menu, choose **Download as**.

Using the AWS CLI

The following `get-object` example command shows how you can use the AWS CLI to download an object from Amazon S3. This command gets the object *`folder/my_image`* from the bucket

amzn-s3-demo-bucket1. You must include an `outfile`, which is a file name for the downloaded object, such as *my_downloaded_image.jpg*.

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --key folder/
my_image my_downloaded_image.jpg
```

For more information and examples, see get-object in the *AWS CLI Command Reference*.

Using the AWS SDKs

For examples of how to download an object with the AWS SDKs, see Code examples in the *Amazon S3 API Reference*.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Using the REST API

You can use the REST API to retrieve objects from Amazon S3. For more information, see GetObject in the *Amazon Simple Storage Service API Reference*.

Downloading multiple objects

You can download multiple objects by using AWS CloudShell, the AWS CLI, or the AWS SDKs.

Using AWS CloudShell in the AWS Management Console

AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console.

For more information about AWS CloudShell, see What is CloudShell? in the *AWS CloudShell User Guide*.

> **Important**
>
> With AWS CloudShell, your home directory has storage up to 1GB per AWS Region. Therefore you cannot sync buckets with objects totaling over this amount. For more limitations, see Service quotas and restrictions in the *AWS CloudShell User Guide*.

To download objects by using AWS CloudShell

1. Sign in to the AWS Management Console and open the CloudShell console at https://console.aws.amazon.com/cloudshell/.
2. Run the following command to sync objects in your bucket to CloudShell. The following command syncs objects from the bucket named *amzn-s3-demo-bucket1* and creates a folder named *temp* in CloudShell. CloudShell syncs your objects to this folder. To use this command, replace the *user input placeholders* with your own information.

```
aws s3 sync s3://amzn-s3-demo-bucket1 ./temp
```

> **Note**
>
> The `sync` command is not compatible with directory buckets.
> To perform pattern matching to either exclude or include particular objects, you can use the `--exclude "value"` and `--include "value"` parameters with the `sync` command.

3. Run the following command to zip your objects in the folder named *temp* to a file named *temp.zip*.

```
zip temp.zip -r temp/
```

4. Choose **Actions**, and then choose **Download file**.
5. Enter the file name **`temp.zip`** and then choose **Download**.
6. (Optional) Delete the *temp.zip* file and the objects that are synced to the *temp* folder in CloudShell. With AWS CloudShell, you have persistent storage of up to 1 GB for each AWS Region.

   You can use the following example command to delete your `.zip` file and your folder. To use this example command, replace the *user input placeholders* with your own information.

```
rm temp.zip && rm -rf temp/
```

Using the AWS CLI

The following example shows how you can use the AWS CLI to download all of the files or objects under the specified directory or prefix. This command copies all objects from the bucket *amzn-s3-demo-bucket1* to your current directory. To use this example command, use your bucket name in place of *amzn-s3-demo-bucket1*.

```
aws s3 cp s3://amzn-s3-demo-bucket1 . --recursive
```

The following command downloads all of the objects under the prefix *logs* in the bucket *amzn-s3-demo-bucket1* to your current directory. It also uses the `--exclude` and `--include` parameters to copy only objects with the suffix *.log*. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3 cp s3://amzn-s3-demo-bucket1/logs/ . --recursive --exclude "*" --include "*.log"
```

For more information and examples, see cp in the *AWS CLI Command Reference*.

Using the AWS SDKs

For examples of how to download all objects in an Amazon S3 bucket with the AWS SDKs, see Code examples in the *Amazon S3 API Reference*.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Downloading part of an object

You can download part of an object by using the AWS CLI or REST API. To do so, you use additional parameters to specify which part of an object that you want to download.

Using the AWS CLI

The following example command performs a GET request for a range of bytes in the object named *folder/my_data* in the bucket named *amzn-s3-demo-bucket1*. In the request, the byte range must be prefixed with `bytes=`. The partial object is downloaded to the output file named *my_data_range*. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --key folder/my_data --range
 bytes=0-500 my_data_range
```

For more information and examples, see get-object in the *AWS CLI Command Reference*.

For more information about the HTTP Range header, see RFC 9110 on the RFC Editor website.

> **Note**
>
> Amazon S3 doesn't support retrieving multiple ranges of data in a single GET request.

Using the REST API

You can use the `partNumber` and `Range` parameters in the REST API to retrieve object parts from Amazon S3. For more information, see GetObject in the *Amazon Simple Storage Service API Reference*.

Downloading an object from another AWS account

You can use a presigned URL to grant others time-limited access to your objects without updating your bucket policy.

The presigned URL can be entered in a browser or used by a program to download an object. The credentials used by the URL are those of the AWS user who generated the URL. After the URL is created, anyone with the presigned URL can download the corresponding object until the URL expires.

Using a presigned URL in the S3 console

You can use the Amazon S3 console to generate a presigned URL for sharing an object in a general purpose bucket by following these steps. When using the console, the maximum expiration time for a presigned URL is 12 hours from the time of creation.

To generate a presigned URL by using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.

3. In the buckets list, choose the name of the bucket that contains the object that you want a presigned URL for.
4. In the **Objects** list, select the object that you want to create a presigned URL for.
5. On the **Object actions** menu, choose **Share with a presigned URL**.
6. Specify how long you want the presigned URL to be valid.
7. Choose **Create presigned URL**.
8. When a confirmation message appears, the URL is automatically copied to your clipboard. You will see a button to copy the presigned URL if you need to copy it again.
9. To download the object, paste the URL into any browser, and the object will attempt to download.

For more information about presigned URLs and other methods for creating them, see Download and upload objects with presigned URLs.

Downloading archived objects

To reduce your storage costs for infrequently accessed objects, you can *archive* those objects. When you archive an object, it is moved into low-cost storage, which means that you can't access it in real time. To download an archived object, you must first restore it.

You can restore archived objects in minutes or hours, depending on the storage class. You can restore an archived object by using the Amazon S3 console, S3 Batch Operations, the Amazon S3 REST API, the AWS SDKs, and the AWS Command Line Interface (AWS CLI).

For instructions, see Restoring an archived object. After you restore the archived object, you can download it.

Downloading objects based on metadata

You can add preconditions to download an object based on it's metadata using a conditional read request. You can return an object based on it's Entity tag (ETag) or last modified date. This can limit an S3 operation to objects updated since a specified date or only return a specific object version.

You can use conditional writes for GetObject or HeadObject requests.

For more information about conditional requests see, Add preconditions to S3 operations with conditional requests.

Troubleshooting downloading objects

Insufficient permissions or incorrect bucket or AWS Identity and Access Management (IAM) user policies can cause errors when you're trying to download objects from Amazon S3. These problems can often cause Access Denied (403 Forbidden) errors, where Amazon S3 is unable to allow access to a resource.

For common causes of Access Denied (403 Forbidden) errors, see Troubleshoot access denied (403 Forbidden) errors in Amazon S3.

If you're getting a 404 NoSuchKey error when trying to access an object, see How can I troubleshoot the 404 NoSuchKey error from Amazon S3? in the AWS re:Post Knowledge Center.

Checking object integrity in Amazon S3

Amazon S3 provides a range of data protection capabilities throughout an object's storage lifecycle. With Amazon S3, you can use checksum values to verify the integrity of the data that you upload or download. In addition, you can request that another checksum value be calculated for any object that you store in S3.

When uploading, copying, or managing your data, you can choose from several supported checksum algorithms:

- CRC-64/NVME (CRC64NVME)

> **Note**
>
> The CRC64NVME checksum algorithm is the default checksum algorithm used for checksum calculations.

- CRC-32 (CRC32)
- CRC-32C (CRC32C)
- SHA-1 (SHA1)
- SHA-256 (SHA256)
- MD5 (MD5)
- XXHash64 (XXHASH64)
- XXHash3 (XXHASH3)

- XXHash128 (XXHASH128)
- SHA-512 (SHA512)

> **Note**
>
> You can provide a precalculated MD5 checksum using the `x-amz-checksum-md5` header. AWS SDKs do not automatically calculate MD5 checksums. For multipart uploads, the Compute checksum operation in S3 Batch Operations can calculate full object MD5 checksums for objects at rest. The legacy `Content-MD5` header remains available for single part uploads using SSE-S3 encryption.

When you upload an object to S3, you can specify the usage of any of these checksum algorithms. For uploads, all AWS-owned clients calculate a checksum of the object and send it with the upload request. S3 then independently calculates a checksum value of the object on the server-side, and validates it with the provided value before storing the object and checksum value. You can also provide precalculated values for these checksum algorithms when you a perform a single part upload or multipart upload (using the full object checksum type for multipart uploads). To use precalculated values with multiple objects, use the AWS CLI or AWS SDKs.

Alternatively, if you want to verify datasets in S3 without the need to restore or download data, you can use the **Compute checksum** operation with S3 Batch Operations. The **Compute checksum** operation allows you to efficiently verify billions of objects in one job request. When the **Compute checksum** operation is performed, S3 calculates the checksum values for a list of objects, at rest. At the end of the job request, you'll receive an automatically generated integrity report (also known as a completion report) that you can use to confirm that your data set remains intact.

Topics

- Checking object integrity for data uploads in Amazon S3
- Checking object integrity for data at rest in Amazon S3

Checking object integrity for data uploads in Amazon S3

Amazon S3 uses checksum values to verify data integrity during upload and download operations. When you upload data, AWS SDK and the AWS Management Console use your chosen checksum algorithm to compute a checksum value before data transmission. S3 then independently

calculates a checksum of your data and validates it against the provided checksum value. Objects are accepted only after confirming data integrity was maintained during transit. S3 stores both the checksum value as object metadata and the object itself.

To verify object integrity, you can request the checksum value during downloads. This validation works consistently across encryption modes, object sizes, storage classes, and both single part and multipart uploads. To change the checksum algorithm for an upload, you can copy an individual object or use batch copy for multiple objects.

For single part uploads, you can provide checksum values as headers. You can either provide a pre-calculated value or let the AWS SDK calculate one during upload. If the checksum value calculated by S3 matches your provided value, the request is accepted. If the values don't match, the request is rejected.

For multipart uploads, AWS SDKs can automatically create trailing checksums for chunked uploads. When you use a trailing checksum, Amazon S3 generates checksum values for every part using your specified algorithm and appends the checksum value to the end of the chunked upload request. S3 performs the verification and upload in a single pass, improving efficiency. For more information, see see Using trailing checksums.

Using supported checksum algorithms

With Amazon S3, you can choose a checksum algorithm to calculate the checksum values during uploads. The specified checksum algorithm is then stored with your object and can be used to validate data integrity during downloads. You can choose one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) checksum algorithms to calculate the checksum value:

- CRC-64/NVME (CRC64NVME)
- CRC-32 (CRC32)
- CRC-32C (CRC32C)
- SHA-1 (SHA1)
- SHA-256 (SHA256)
- MD5 (MD5)
- XXHash64 (XXHASH64)
- XXHash3 (XXHASH3)

- XXHash128 (XXHASH128)
- SHA-512 (SHA512)

> **Note**
>
> The `content-MD5` header is only available using the S3 ETag for objects uploaded in a single part upload (PUT operation) that uses the SSE-S3 encryption.

Additionally, you can provide a checksum with each request using the Content-MD5 header.

When you upload an object, you specify the algorithm that you want to use:

- **When you use the AWS Management Console**, choose the checksum algorithm that you want to use. You can optionally specify the checksum value of the object. When Amazon S3 receives the object, it calculates the checksum by using the algorithm that you specified. If the two checksum values don't match, Amazon S3 generates an error.
- **When you use an SDK**, be aware of the following:
  - Set the `ChecksumAlgorithm` parameter to the algorithm that you want Amazon S3 to use. If you already have a precalculated checksum, you pass the checksum value to the AWS SDK, and the SDK includes the value in the request. If you don’t pass a checksum value or don’t specify a checksum algorithm, the SDK automatically calculates a checksum value for you and includes it with the request to provide integrity protections. If the individual checksum value doesn't match the set value of the checksum algorithm, Amazon S3 fails the request with a `BadDigest` error.
  - If you’re using an upgraded AWS SDK, the SDK chooses a checksum algorithm for you. However, you can override this checksum algorithm.
  - If you don’t specify a checksum algorithm and the SDK also doesn’t calculate a checksum for you, then S3 automatically chooses the CRC-64/NVME (`CRC64NVME`) checksum algorithm.
- **When you use the REST API**, don't use the `x-amz-sdk-checksum-algorithm` parameter. Instead, use one of the algorithm-specific headers (for example, `x-amz-checksum-crc32`).

To apply any of these checksum values to objects that are already uploaded to Amazon S3, you can copy the object and specify whether you want to use the existing checksum algorithm or a new one. If you don’t specify an algorithm, S3 uses the existing algorithm. If the source object

doesn't have a specified checksum algorithm or checksum value, Amazon S3 uses the CRC-64/NVME algorithm to calculate the checksum value for the destination object. You can also specify a checksum algorithm when copying objects using S3 Batch Operations.

> **Important**
>
> If you use a multipart upload with **Checksums** for composite (or part-level) checksums, the multipart upload part numbers must be consecutive and begin with 1. If you try to complete a multipart upload request with nonconsecutive part numbers, Amazon S3 generates an `HTTP 500 Internal Server` error.

Full object and composite checksum types

In Amazon S3, there are two types of supported checksums:

- **Full object checksums:** A full object checksum is calculated based on all of the content of a multipart upload, covering all data from the first byte of the first part to the last byte of the last part. Be aware that if you're using the AWS Management Console to upload objects that are smaller than 16 MB, only the full object checksum type is supported.

  > **Note**
  >
  > Objects that you upload using `PutObject` use the full object checksum type. You do not need to specify a checksum type.

- **Composite checksums:** A composite checksum is calculated based on the individual checksums of each part in a multipart upload. Instead of computing a checksum based on all of the data content, this approach aggregates the part-level checksums (from the first part to the last) to produce a single, combined checksum for the complete object. You may only use the composite checksum type with certain checksum algorithms and only when using a multipart upload to upload your object. For the algorithms that support each checksum type, see the Multipart uploads section.

  > **Note**
  >
  > When an object is uploaded as a multipart upload, the entity tag (ETag) for the object is not an MD5 digest of the entire object. Instead, Amazon S3 calculates the MD5 digest of each individual part as it is uploaded. The MD5 digests are used to determine the ETag

for the final object. Amazon S3 concatenates the bytes for the MD5 digests together and then calculates the MD5 digest of these concatenated values. During the final ETag creation step, Amazon S3 adds a dash with the total number of parts to the end.

Single part uploads

Checksums of objects that are uploaded in a single part (using `PutObject`) are treated as full object checksums. When you upload an object in the Amazon S3 console, you can choose the checksum algorithm that you want S3 to use and also (optionally) provide a precomputed value. Amazon S3 then validates the precomputed checksum value before storing the object and its checksum value. You can verify an object's data integrity when you request the checksum value during object downloads.

Multipart uploads

When you upload the object in multiple parts using the `MultipartUpload` API, you can specify the checksum algorithm that you want Amazon S3 to use and the checksum type (full object or composite).

The following table indicates which checksum algorithm type is supported for each checksum algorithm in a multipart upload:

Checksum algorithm	Full object	Composite
CRC-64/NVME (CRC64NVME)	Yes	No
CRC-32 (CRC32)	Yes	Yes
CRC-32C (CRC32C)	Yes	Yes
SHA-1 (SHA1)	No	Yes
SHA-256 (SHA256)	No	Yes
MD5 (MD5)	No	Yes
XXHash64 (XXHASH64)	No	Yes
XXHash3 (XXHASH3)	No	Yes

Checksum algorithm	Full object	Composite
XXHash128 (XXHASH128)	No	Yes
SHA-512 (SHA512)	No	Yes

Using full object checksums for multipart upload

When creating or performing a multipart upload, you can use full object checksums for validation on upload. This means that you can provide the checksum algorithm for the `MultipartUpload` API, simplifying your integrity validation tooling because you no longer need to track part boundaries for uploaded objects. You can provide the checksum of the whole object in the `CompleteMultipartUpload` request, along with the object size.

When you provide a full object checksum during a multipart upload, the AWS SDK passes the checksum to Amazon S3, and S3 validates the object integrity server-side, comparing it to the received value. Then, Amazon S3 stores the object if the values match. If the two values don't match, S3 fails the request with a `BadDigest` error. The checksum of your object is also stored in object metadata that you use later to validate an object's data integrity.

For full object checksums, you can use CRC-64/NVME (`CRC64NVME`), CRC-32 (`CRC32`), or CRC-32C (`CRC32C`) checksum algorithms in S3. Full object checksums in multipart uploads are only available for CRC-based checksums because they can linearize into a full object checksum. This linearization allows Amazon S3 to parallelize your requests for improved performance. In particular, S3 can compute the checksum of the whole object from the part-level checksums. This type of validation isn't available for other algorithms, such as SHA and MD5. Because S3 has default integrity protections, if objects are uploaded without a checksum, S3 automatically attaches the recommended full object CRC-64/NVME (`CRC64NVME`) checksum algorithm to the object.

> **Note**
>
> To initiate the multipart upload, you can specify the checksum algorithm and the full object checksum type. After you specify the checksum algorithm and the full object checksum type, you can provide the full object checksum value for the multipart upload.

Important

When using multipart uploads with the new checksum algorithms (MD5, XXHash3, XXHash64, XXHash128, SHA-512), you must specify the checksum algorithm in the `CreateMultipartUpload` request using the `x-amz-checksum-algorithm` header. If you provide a checksum value in the `CompleteMultipartUpload` request (for example, `x-amz-checksum-sha512`) without having specified the algorithm in `CreateMultipartUpload`, the request will fail with an `InvalidRequest` error.

Note

For existing checksum algorithms (CRC32, CRC32C, SHA-1, SHA-256), if the algorithm is not specified in `CreateMultipartUpload`, any checksum header provided in `CompleteMultipartUpload` is currently accepted but not validated or stored with the object. We recommend always specifying the algorithm in `CreateMultipartUpload` to ensure your checksums are validated and stored.

Using part-level checksums for multipart upload

When objects are uploaded to Amazon S3, they can be uploaded as a single object or uploaded in parts with the multipart upload process. You can choose a **Checksum** type for your multipart upload. For multipart upload part-level checksums (or composite checksums), Amazon S3 calculates the checksum for each individual part by using the specified checksum algorithm. You can use `UploadPart` to provide the checksum values for each part. If the object that you try to upload in the Amazon S3 console is set to use the CRC-64/NVME (CRC64NVME) checksum algorithm and exceeds 16 MB, it is automatically designated as a full object checksum.

Amazon S3 then uses the stored part-level checksum values to confirm that each part is uploaded correctly. When each part's checksum (for the whole object) is provided, S3 uses the stored checksum values of each part to calculate the full object checksum internally, comparing it with the provided checksum value. This minimizes compute costs since S3 can compute a checksum of the whole object using the checksum of the parts. For more information about multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3 and Using full object checksums for multipart upload.

When the object is completely uploaded, you can use the final calculated checksum to verify the data integrity of the object.

When uploading a part of the multipart upload, be aware of the following:

- To retrieve information about the object, including how many parts make up the entire object, you can use the `GetObjectAttributes` operation. With additional checksums, you can also recover information for each individual part that includes the part's checksum value.
- For completed uploads, you can get an individual part's checksum by using the `GetObject` or `HeadObject` operations and specifying a part number or byte range that aligns with a single part. If you want to retrieve the checksum values for individual parts of multipart uploads that are still in progress, you can use `ListParts`.
- Because of how Amazon S3 calculates the checksum for multipart objects, the checksum value for the object might change if you copy it. If you're using an SDK or the REST API and you call `CopyObject`, Amazon S3 copies any object up to the size limitations of the `CopyObject` API operation. Amazon S3 does this copy as a single action, regardless of whether the object was uploaded in a single request or as part of a multipart upload. With a copy command, the checksum of the object is a direct checksum of the full object. If the object was originally uploaded using a multipart upload, the checksum value changes even though the data doesn't.
- Objects that are larger than the size limitations of the `CopyObject` API operation must use multipart upload copy commands.
- When you perform some operations using the AWS Management Console, Amazon S3 uses a multipart upload if the object is greater than 16 MB in size.

Checksum methods

After uploading objects, you can get the checksum value and compare it to a precomputed or previously stored checksum value of the same checksum algorithm type. The following examples show you which checksum calculation methods you can use to verify data integrity.

Using the S3 console

To learn more about using the console and specifying checksum algorithms to use when uploading objects, see Uploading objects and Tutorial: Checking the integrity of data in Amazon S3 with additional checksums.

Using the AWS SDKs

The following example shows how you can use the AWS SDKs to upload a large file with multipart upload, download a large file, and validate a multipart upload file, all by using SHA-256 for file validation.

Java

Example: Uploading, downloading, and verifying a large file with SHA-256

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import software.amazon.awssdk.auth.credentials.AwsCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.core.ResponseInputStream;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.ChecksumMode;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.PutObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.Tag;
import software.amazon.awssdk.services.s3.model.Tagging;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
```

```
    import java.io.IOException;
    import java.io.InputStream;
    import java.io.OutputStream;
    import java.nio.ByteBuffer;
    import java.security.MessageDigest;
    import java.security.NoSuchAlgorithmException;
    import java.util.ArrayList;
    import java.util.Base64;
    import java.util.List;

    public class LargeObjectValidation {
        private static String FILE_NAME = "sample.file";
        private static String BUCKET = "sample-bucket";
        //Optional, if you want a method of storing the full multipart object
 checksum in S3.
        private static String CHECKSUM_TAG_KEYNAME = "fullObjectChecksum";
        //If you have existing full-object checksums that you need to validate
 against, you can do the full object validation on a sequential upload.
        private static String SHA256_FILE_BYTES = "htCM5g7ZNdoSw8bN/
mkgiAhXt5MFoVowVg+LE9aIQmI=";
        //Example Chunk Size - this must be greater than or equal to 5MB.
        private static int CHUNK_SIZE = 5 * 1024 * 1024;

        public static void main(String[] args) {
            S3Client s3Client = S3Client.builder()
                    .region(Region.US_EAST_1)
                    .credentialsProvider(new AwsCredentialsProvider() {
                        @Override
                        public AwsCredentials resolveCredentials() {
                            return new AwsCredentials() {
                                @Override
                                public String accessKeyId() {
                                    return Constants.ACCESS_KEY;
                                }

                                @Override
                                public String secretAccessKey() {
                                    return Constants.SECRET;
                                }
                            };
                        }
                    })
                    .build();
            uploadLargeFileBracketedByChecksum(s3Client);
```

```
            downloadLargeFileBracketedByChecksum(s3Client);
            validateExistingFileAgainstS3Checksum(s3Client);
        }

        public static void uploadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting uploading file validation");
            File file = new File(FILE_NAME);
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256 = MessageDigest.getInstance("SHA-256");
                CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
                        .bucket(BUCKET)
                        .key(FILE_NAME)
                        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
                        .build();
                CreateMultipartUploadResponse createdUpload =
s3Client.createMultipartUpload(createMultipartUploadRequest);
                List<CompletedPart> completedParts = new ArrayList<CompletedPart>();
                int partNumber = 1;
                byte[] buffer = new byte[CHUNK_SIZE];
                int read = in.read(buffer);
                while (read != -1) {
                    UploadPartRequest uploadPartRequest =
UploadPartRequest.builder()

.partNumber(partNumber).uploadId(createdUpload.uploadId()).key(FILE_NAME).bucket(BUCKET).ch
                    UploadPartResponse uploadedPart =
s3Client.uploadPart(uploadPartRequest,
RequestBody.fromByteBuffer(ByteBuffer.wrap(buffer, 0, read)));
                    CompletedPart part =
CompletedPart.builder().partNumber(partNumber).checksumSHA256(uploadedPart.checksumSHA256()
                    completedParts.add(part);
                    sha256.update(buffer, 0, read);
                    read = in.read(buffer);
                    partNumber++;
                }
                String fullObjectChecksum =
Base64.getEncoder().encodeToString(sha256.digest());
                if (!fullObjectChecksum.equals(SHA256_FILE_BYTES)) {
                    //Because the SHA256 is uploaded after the part is uploaded; the
upload is bracketed and the full object can be fully validated.

s3Client.abortMultipartUpload(AbortMultipartUploadRequest.builder().bucket(BUCKET).key(FILE
```

```
                    throw new IOException("Byte mismatch between stored checksum and
upload, do not proceed with upload and cleanup");
                }
                CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder().parts(completedParts).build();
                CompleteMultipartUploadResponse completedUploadResponse =
s3Client.completeMultipartUpload(

CompleteMultipartUploadRequest.builder().bucket(BUCKET).key(FILE_NAME).uploadId(createdUplo
                Tag checksumTag =
Tag.builder().key(CHECKSUM_TAG_KEYNAME).value(fullObjectChecksum).build();
                //Optionally, if you need the full object checksum stored with the
file; you could add it as a tag after completion.

s3Client.putObjectTagging(PutObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).t
            } catch (IOException | NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
            GetObjectAttributesResponse
                    objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
            System.out.println(objectAttributes.objectParts().parts());
            System.out.println(objectAttributes.checksum().checksumSHA256());
        }

        public static void downloadLargeFileBracketedByChecksum(S3Client s3Client) {
            System.out.println("Starting downloading file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            try (OutputStream out = new FileOutputStream(file)) {
                GetObjectAttributesResponse
                        objectAttributes =
s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                        .objectAttributes(ObjectAttributes.OBJECT_PARTS,
ObjectAttributes.CHECKSUM).build());
                //Optionally if you need the full object checksum, you can grab a
tag you added on the upload
                List<Tag> objectTags =
s3Client.getObjectTagging(GetObjectTaggingRequest.builder().bucket(BUCKET).key(FILE_NAME).b
                String fullObjectChecksum = null;
                for (Tag objectTag : objectTags) {
                    if (objectTag.key().equals(CHECKSUM_TAG_KEYNAME)) {
                        fullObjectChecksum = objectTag.value();
```

```
                        break;
                    }
                }
                MessageDigest sha256FullObject =
 MessageDigest.getInstance("SHA-256");
                MessageDigest sha256ChecksumOfChecksums =
 MessageDigest.getInstance("SHA-256");

                //If you retrieve the object in parts, and set the ChecksumMode to
 enabled, the SDK will automatically validate the part checksum
                for (int partNumber = 1; partNumber <=
 objectAttributes.objectParts().totalPartsCount(); partNumber++) {
                    MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
                    ResponseInputStream<GetObjectResponse> response =
 s3Client.getObject(GetObjectRequest.builder().bucket(BUCKET).key(FILE_NAME).partNumber(part
                    GetObjectResponse getObjectResponse = response.response();
                    byte[] buffer = new byte[CHUNK_SIZE];
                    int read = response.read(buffer);
                    while (read != -1) {
                        out.write(buffer, 0, read);
                        sha256FullObject.update(buffer, 0, read);
                        sha256Part.update(buffer, 0, read);
                        read = response.read(buffer);
                    }
                    byte[] sha256PartBytes = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(sha256PartBytes);
                    //Optionally, you can do an additional manual validation again
 the part checksum if needed in addition to the SDK check
                    String base64PartChecksum =
 Base64.getEncoder().encodeToString(sha256PartBytes);
                    String base64PartChecksumFromObjectAttributes =
 objectAttributes.objectParts().parts().get(partNumber - 1).checksumSHA256();
                    if (!
base64PartChecksum.equals(getObjectResponse.checksumSHA256()) || !
base64PartChecksum.equals(base64PartChecksumFromObjectAttributes)) {
                        throw new IOException("Part checksum didn't match for the
 part");
                    }
                    System.out.println(partNumber + " " + base64PartChecksum);
                }
                //Before finalizing, do the final checksum validation.
                String base64FullObject =
 Base64.getEncoder().encodeToString(sha256FullObject.digest());
```

```
                String base64ChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                if (fullObjectChecksum != null && !
fullObjectChecksum.equals(base64FullObject)) {
                    throw new IOException("Failed checksum validation for full
 object");
                }
                System.out.println(fullObjectChecksum);
                String base64ChecksumOfChecksumFromAttributes =
 objectAttributes.checksum().checksumSHA256();
                if (base64ChecksumOfChecksumFromAttributes != null && !
base64ChecksumOfChecksums.equals(base64ChecksumOfChecksumFromAttributes)) {
                    throw new IOException("Failed checksum validation for full
 object checksum of checksums");
                }
                System.out.println(base64ChecksumOfChecksumFromAttributes);
                out.flush();
            } catch (IOException | NoSuchAlgorithmException e) {
                //Cleanup bad file
                file.delete();
                e.printStackTrace();
            }
        }

        public static void validateExistingFileAgainstS3Checksum(S3Client s3Client)
 {
            System.out.println("Starting existing file validation");
            File file = new File("DOWNLOADED_" + FILE_NAME);
            GetObjectAttributesResponse
                    objectAttributes =
 s3Client.getObjectAttributes(GetObjectAttributesRequest.builder().bucket(BUCKET).key(FILE_N
                    .objectAttributes(ObjectAttributes.OBJECT_PARTS,
 ObjectAttributes.CHECKSUM).build());
            try (InputStream in = new FileInputStream(file)) {
                MessageDigest sha256ChecksumOfChecksums =
 MessageDigest.getInstance("SHA-256");
                MessageDigest sha256Part = MessageDigest.getInstance("SHA-256");
                byte[] buffer = new byte[CHUNK_SIZE];
                int currentPart = 0;
                int partBreak =
 objectAttributes.objectParts().parts().get(currentPart).size();
                int totalRead = 0;
                int read = in.read(buffer);
                while (read != -1) {
```

```
                    totalRead += read;
                    if (totalRead >= partBreak) {
                        int difference = totalRead - partBreak;
                        byte[] partChecksum;
                        if (totalRead != partBreak) {
                            sha256Part.update(buffer, 0, read - difference);
                            partChecksum = sha256Part.digest();
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                            sha256Part.update(buffer, read - difference,
 difference);
                        } else {
                            sha256Part.update(buffer, 0, read);
                            partChecksum = sha256Part.digest();
                            sha256ChecksumOfChecksums.update(partChecksum);
                            sha256Part.reset();
                        }
                        String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                        if (!
base64PartChecksum.equals(objectAttributes.objectParts().parts().get(currentPart).checksumSH
 {
                            throw new IOException("Part checksum didn't match S3");
                        }
                        currentPart++;
                        System.out.println(currentPart + " " + base64PartChecksum);
                        if (currentPart <
 objectAttributes.objectParts().totalPartsCount()) {
                            partBreak +=
 objectAttributes.objectParts().parts().get(currentPart - 1).size();
                        }
                    } else {
                        sha256Part.update(buffer, 0, read);
                    }
                    read = in.read(buffer);
                }
                if (currentPart != objectAttributes.objectParts().totalPartsCount())
 {
                    currentPart++;
                    byte[] partChecksum = sha256Part.digest();
                    sha256ChecksumOfChecksums.update(partChecksum);
                    String base64PartChecksum =
 Base64.getEncoder().encodeToString(partChecksum);
                    System.out.println(currentPart + " " + base64PartChecksum);
```

```
                }

                String base64CalculatedChecksumOfChecksums =
 Base64.getEncoder().encodeToString(sha256ChecksumOfChecksums.digest());
                System.out.println(base64CalculatedChecksumOfChecksums);
                System.out.println(objectAttributes.checksum().checksumSHA256());
                if (!
base64CalculatedChecksumOfChecksums.equals(objectAttributes.checksum().checksumSHA256()))
 {
                    throw new IOException("Full object checksum of checksums don't
 match S3");
                }

            } catch (IOException | NoSuchAlgorithmException e) {
                e.printStackTrace();
            }
        }
    }
```

Using the REST API

You can send REST requests to upload an object with a checksum value to verify the integrity of the data with PutObject. You can also retrieve the checksum value for objects using GetObject or HeadObject.

Using the AWS CLI

You can send a PUT request to upload an object of up to 5 GB in a single operation. For more information, see the `PutObject` in the *AWS CLI Command Reference*. You can also use `get-object` and `head-object` to retrieve the checksum of an already-uploaded object to verify the integrity of the data.

For information, see Amazon S3 CLI FAQ in the *AWS Command Line Interface User Guide*.

Using Content-MD5 when uploading objects

Another way to verify the integrity of your object after uploading is to provide an MD5 digest of the object when you upload it. If you calculate the MD5 digest for your object, you can provide the digest with the PUT command by using the `Content-MD5` header.

After uploading the object, Amazon S3 calculates the MD5 digest of the object and compares it to the value that you provided. The request succeeds only if the two digests match.

Supplying an MD5 digest isn't required, but you can use it to verify the integrity of the object as part of the upload process.

Using Content-MD5 and the ETag to verify uploaded objects

The entity tag (ETag) for an object represents a specific version of that object. Keep in mind that the ETag only reflects changes to the content of an object, not changes to its metadata. If only the metadata of an object changes, the ETag remains the same.

Depending on the object, the ETag of the object might be an MD5 digest of the object data:

- If an object is created by the `PutObject`, `PostObject`, or `CopyObject` operation, or through the AWS Management Console, and that object is also plaintext or encrypted by server-side encryption with Amazon S3 managed keys (SSE-S3), that object has an ETag that is an MD5 digest of its object data.
- If an object is created by the `PutObject`, `PostObject`, or `CopyObject` operation, or through the AWS Management Console, and that object is encrypted by server-side encryption with customer-provided keys (SSE-C) or server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), that object has an ETag that is not an MD5 digest of its object data.
- If an object is created by either the multipart upload process or the `UploadPartCopy` operation, the object's ETag is not an MD5 digest, regardless of the method of encryption. If an object is larger than 16 MB, the AWS Management Console uploads or copies that object as a multipart upload, and therefore the ETag isn't an MD5 digest.

For objects where the ETag is the `Content-MD5` digest of the object, you can compare the ETag value of the object with a calculated or previously stored `Content-MD5` digest.

Using trailing checksums

When uploading large objects to Amazon S3, you can either provide a precalculated checksum for the object or use an AWS SDK to automatically create trailing checksums for chunked uploads, on your behalf. If you use a trailing checksum, Amazon S3 automatically generates the checksum value by using your specified algorithm to validate the integrity of the object in chunked uploads, when you upload an object.

To create a trailing checksum when using an AWS SDK, populate the `ChecksumAlgorithm` parameter with your preferred algorithm. The SDK uses that algorithm to calculate the checksum value for your object (or object parts) and automatically appends it to the end of your chunked

upload request. This behavior saves you time because Amazon S3 performs both the verification and upload of your data in a single pass.

> **Important**
>
> If you're using S3 Object Lambda, all requests to S3 Object Lambda are signed using `s3-object-lambda` instead of `s3`. This behavior affects the signature of trailing checksum values. For more information about S3 Object Lambda, see Transforming objects with S3 Object Lambda.

Trailing checksum headers

To make a chunked content encoding request, Amazon S3 requires client servers to include several headers to correctly parse the request. Client servers must include the following headers:

- **`x-amz-decoded-content-length`:**This header indicates the plaintext size of the actual data that is being uploaded to Amazon S3 with the request.
- **`x-amz-content-sha256`:** This header indicates the type of chunked upload that is included in the request. For chunked uploads with trailing checksums, the header value is `STREAMING-UNSIGNED-PAYLOAD-TRAILER` for requests that don't use payload signing and `STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER` for requests that use SigV4 payload signing. (For more information about implementing signed payloads, see Signature calculations for the authorization header: Transferring a payload in multiple chunks.)
- **`x-amz-trailer`:** This header indicates the name of the trailing header in the request. If trailing checksums exist (where AWS SDKs append checksums to the encoded request bodies), the `x-amz-trailer` header value includes the `x-amz-checksum-` prefix and ends with the algorithm name. The following `x-amz-trailer` values are currently supported:
  - `x-amz-checksum-crc32`
  - `x-amz-checksum-crc32c`
  - `x-amz-checksum-crc64nvme`
  - `x-amz-checksum-sha1`
  - `x-amz-checksum-sha256`

> **Note**
>
> You can also include the `Content-Encoding` header, with the chunked value, in your request. While this header isn't required, including this header can minimize HTTP proxy issues when transmitting encoded data. If another `Content-Encoding` header (such as gzip) exists in the request, the `Content-Encoding` header includes the chunked value in a comma-separated list of encodings. For example, `Content-Encoding: aws-chunked, gzip`.

Chunked parts

When you upload an object to Amazon S3 using chunked encoding, the upload request includes the following types of chunks (formatted in the listed order):

- **Object body chunks:** There can be one, multiple, or zero body chunks associated with a chunked upload request.
- **Completion chunks:** There can be one, multiple, or zero body chunks associated with a chunked upload request.
- **Trailing chunks:** The trailing checksum is listed after the completion chunk. Only one trailing chunk is allowed.

> **Note**
>
> Every chunked upload must end with a final CRLF (such as `\r\n`) to indicate the end of the request.

For examples of chunked formatting, see Examples: Chunked uploads with trailing checksums.

Object body chunks

Object body chunks are the chunks that contain the actual object data that is being uploaded to S3. These chunks have consistent size and format constraints.

Object body chunk size

These chunks must contain at least 8,192 bytes (or 8 KiB) of object data, except for the final body chunk, which can be smaller. There is no explicit maximum chunk size but you can expect all chunks

to be smaller than the 5 GB maximum upload size. Chunk sizes can vary from one chunk to the next based on your client server implementation.

Object body chunk format

Object body chunks begin with the hexadecimal encoding of the number of bytes in the object body chunk, followed by a CRLF (Carriage Return Line Feed), the object bytes for that chunk, and another CRLF.

For example:

```
hex-encoding-of-object-bytes-in-chunk\r\n
chunk-object-bytes\r\n
```

However, when the chunk is signed, the object body chunk follows a different format, where the signature is appended to the chunk size with a semicolon delimiter. For example:

```
hex-encoding-of-object-bytes-in-chunk;chunk-signature\r\n
 chunk-object-bytes\r\n
```

For more information about chunk signing, see Signature calculations for the Authorization Header: Transferring a payload in multiple chunks (AWS Signature Version 4). For more information about chunk formatting, see Chunked transfer encoding on the *RFC Editor* website.

Completion chunks

Completion chunks are required to be the final object body chunk of every chunked upload. A completion chunk's format is similar to a body chunk, but always contains zero bytes of object data. (The zero bytes of object data indicate that all of the data has been uploaded.) Chunked uploads must include a completion chunk as its final object body chunk, following a format like this:

```
0\r\n
```

However, if the content encoding request uses payload signing, it follows this format instead:

```
0;chunk-signature\r\n
```

Trailer chunks

Trailer chunks hold the calculated checksum for all S3 upload requests. Trailer chunks include two fields: one header name field and one header value field. The header name field for an upload request must match the value passed into the `x-amz-trailer` request header. For example, if a request contains `x-amz-trailer: x-amz-checksum-crc32` and the trailer chunk has the header name `x-amz-checksum-sha1`, the request fails. The value field in the trailer chunk includes a base64 encoding of the big-endian checksum value for that object. (The big-endian ordering stores the most significant byte of data at the lowest memory address, and the least significant byte at the largest memory address). The algorithm used to calculate this checksum is the same as the suffix for the header name (for example, `crc32`).

Trailer chunk format

Trailer chunks use the following format for unsigned payload requests:

```
x-amz-checksum-lowercase-checksum-algorithm-name:base64-checksum-value\n\r\n\r\n
```

For requests with SigV4 signed payloads, the trailer chunk includes a trailer signature after the trailer chunk.

```
trailer-checksum\n\r\n
trailer-signature\r\n
```

You can also add the CRLF directly to the end of the base64 checksum value. For example:

```
x-amz-checksum-lowercase-checksum-algorithm-name:base64-checksum-value\r\n\r\n
```

Examples: Chunked uploads with trailing checksums

Amazon S3 supports chunked uploads that use `aws-chunked` content encoding for `PutObject` and `UploadPart` requests with trailing checksums.

Example 1 – Unsigned chunked `PutObject` request with a trailing CRC-32 checksum

The following is an example of a chunked `PutObject` request with a trailing CRC-32 checksum. In this example, the client uploads a 17 KB object in three unsigned chunks and appends a trailing CRC-32 checksum chunk by using the `x-amz-checksum-crc32` header.

```
PUT /Key+ HTTP/1.1
```

```
Host: amzn-s3-demo-bucket
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 17408
x-amz-content-sha256: STREAMING-UNSIGNED-PAYLOAD-TRAILER
x-amz-trailer: x-amz-checksum-crc32

2000\r\n                                    // Object body chunk 1 (8192 bytes)
object-bytes\r\n
2000\r\n                                    // Object body chunk 2 (8192 bytes)
object-bytes\r\n
400\r\n                                     // Object body chunk 3 (1024 bytes)
object-bytes\r\n
0\r\n                                       // Completion chunk
x-amz-checksum-crc32:YABb/g==\n\r\n\r\n     // Trailer chunk (note optional \n
 character)
\r\n                                          // CRLF
```

Here's an example response:

```
HTTP/1.1 200
ETag: ETag
x-amz-checksum-crc32: YABb/g==
```

> **Note**
>
> The usage of the linefeed \n at the end of the checksum value might vary across clients.

Example 2 – SigV4-signed chunked `PutObject` request with a trailing CRC-32 (CRC32) checksum

The following is an example of a chunked `PutObject` request with a trailing CRC-32 checksum. This request uses SigV4 payload signing. In this example, the client uploads a 17 KB object in three signed chunks. In addition to the `object body` chunks, the `completion chunk` and `trailer` chunk are also signed.

```
PUT /Key+ HTTP/1.1
Host: amzn-s3-demo-bucket.s3.amazonaws.com
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 17408
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
```

```
x-amz-trailer: x-amz-checksum-crc32

authorization-code                                  // SigV4 headers authorization

2000;chunk-signature=signature-value...\r\n   // Object body chunk 1 (8192 bytes)
object-bytes\r\n
2000;chunk-signature\r\n                            // Object body chunk 2 (8192 bytes)
object-bytes\r\n
400;chunk-signature\r\n                             // Object body chunk 3 (1024 bytes)
object-bytes\r\n
0;chunk-signature\r\n                               // Completion chunk
x-amz-checksum-crc32:YABb/g==\n\r\n                  // Trailer chunk (note optional \n
 character)
trailer-signature\r\n
\r\n                                                // CRLF
```

Here's an example response:

```
HTTP/1.1 200
ETag: ETag
x-amz-checksum-crc32: YABb/g==
```

Checking object integrity for data at rest in Amazon S3

If you need to verify the content of datasets stored in Amazon S3, the S3 Batch Operations Compute checksum operation calculates both full object or composite checksums for objects at rest. The **Compute checksum** operation uses Batch Operations to asynchronously calculate the checksum values for a group of objects and automatically generates a consolidated integrity report, without creating new copies of your data, or restoring or downloading any data.

With the **Compute checksum** operation, you can efficiently verify billions of objects with a single job request. For each **Compute checksum** job request, S3 calculates the checksum values, and includes it in an automatically generated integrity report (also known as a completion report). You can then use the completion report to validate the integrity of your dataset.

The **Compute checksum** operation works with any object stored in S3, regardless of storage class or object size. Whether you need to verify your objects as a data preservation best practice, or meet compliance requirements, the **Compute checksum** operation reduces the cost, time, and effort required for data validation by performing checksum calculations at rest. For information about **Compute checksum** pricing, see the **Management and insights** tab on Amazon S3 pricing.

Then, you can use the output of the generated completion report to compare against the checksum values that you've stored in your databases to verify that your datasets remain intact over time. This approach helps you maintain end-to-end data integrity for business and compliance needs. For example, you can use the **Compute checksum** operation to submit a list of stored objects in S3 Glacier storage classes for annual security audits. Additionally, the range of supported checksum algorithms allow you to maintain continuity with the algorithms that are used in your applications.

Using supported checksum algorithms

For data at rest, you can calculate checksums, using any of the supported checksum algorithms:

All supported checksum algorithms support both the full object and composite checksum types, except CRC-64/NVME (CRC64NVME), which supports the full object checksum type only.

- CRC-64/NVME (CRC64NVME)
- CRC-32 (CRC32)
- CRC-32C (CRC32C)
- SHA-1 (SHA1)
- SHA-256 (SHA256)
- MD5 (MD5)
- XXHash64 (XXHASH64)
- XXHash3 (XXHASH3)
- XXHash128 (XXHASH128)
- SHA-512 (SHA512)

Using Compute checksum

For objects stored in Amazon S3, you can use the **Compute checksum** operation with S3 Batch Operations to check the content of stored data at rest. You can create a Compute checksum Batch Operations job by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), REST API, or AWS SDK. When the **Compute checksum** job finishes, you receive a completion report. For more information about how to use the completion report, see Tracking job status and completion reports.

Before creating your **Compute checksum** job, you must create an S3 Batch Operations AWS Identity and Access Management (IAM) role to grant Amazon S3 permissions to perform actions on

your behalf. You'll need to grant permissions to read the manifest file and to write a completion report to the S3 bucket. For more information, see Compute checksums.

Using the S3 console

To use the Compute checksum operation

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create your job.

   > **Note**
   >
   > For copy operations, you must create the job in the same Region as the destination bucket. For all other operations, you must create the job in the same Region as the objects in the manifest.

3. Choose **Batch Operations** on the left navigation pane of the Amazon S3 console.
4. Choose **Create job**.
5. View the AWS Region where you want to create your job.

   > **Note**
   >
   > For copy operations, you must create the job in the same Region as the destination bucket. For all other operations, you must create the job in the same Region as the objects in the manifest.

6. Under **Manifest format**, choose the type of manifest object to use.

   - If you choose **S3 inventory report (manifest.json)**, enter the path to the `manifest.json` object, and (optionally) the **Manifest object version ID** if you want to use a specific object version. Alternatively, you can choose **Browse S3** and choose the manifest JSON file, which auto populates all the manifest object field entries.
   - If you choose **CSV**, choose the Manifest location type, Then, either enter the path to a CSV-formatted manifest object or choose **Browse S3** to select a manifest object. The manifest object must follow the format described in the console. If you want to use a specific version of the manifest object, then you can also specify the object version ID.

- If you choose **Create manifest using S3 Replication configuration**, a list of objects will be generated using the replication configuration and optionally saved to the destination you choose. When using a replication configuration to generate the manifest, the only operation that will be available is **Replicate**.

7. Choose **Next**.
8. Under **Operation**, choose the **Compute checksum** operation to calculate the checksums on all objects listed in the manifest. Choose the **Checksum type** and **Checksum function** for your job. Then, choose **Next**.
9. Fill out the information for **Configure additional options**, and then choose **Next**.
10. On the **Configure additional options** page, fill out the information for your **Compute checksum** job.

> **Note**
>
> Under **Completion report**, make sure to confirm the acknowledgement statement. This acknowledgement statement confirms that you understand that the completion report contains checksum values, which are used to verify the integrity of data stored in Amazon S3. Therefore, the completion report should be shared with caution. Also, be aware that if you're creating a Compute checksum request and you specify an external account owner's bucket location to store your completion report, make sure to specify the AWS account ID of the external bucket owner.

11. Choose **Next**.
12. On the **Review** page, review and confirm your settings.
13. (Optional) If you need to make changes, choose **Previous** to go back to the previous page, or choose **Edit** to update a specific step.
14. After you've confirmed your changes, choose **Create job**.

To list and monitor the progress of all Compute checksum requests

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. On the **Batch Operations** page, you can review job details such as the job priority, job completion rate, and total objects.

4. If you want to manage or clone a specific **Compute checksum** job, choose the **Job ID** to review additional job information.
5. On the specific **Compute checksum** job page, review the job details.

Each batch operations job progresses through different job statuses. You can also enable AWS CloudTrail events in the S3 console to receive alerts on any job state changes. For active jobs, you can review the running job and completion rate on the **Job details** page.

Using the AWS SDKs

Java

Example: Creating a Compute checksum job

The following example shows you how to create a **Compute checksum** job (as part of a **Create job** request), and how to specify a manifest:

```
// Required parameters
String accountId = "111122223333";
String roleArn = "arn:aws:iam::111122223333:role/BatchOperations";
String manifestArn = "arn:aws:s3:::my_manifests/manifest.csv";
String manifestEtag = "60e460c9d1046e73f7dde5043ac3ae85";
String reportBucketArn = "arn:aws:s3:::amzn-s3-demo-completion-report-bucket";
String reportExpectedBucketOwner = "111122223333";
String reportPrefix = "demo-report";

// Job Operation
S3ComputeObjectChecksumOperation s3ComputeObjectChecksum =
 S3ComputeObjectChecksumOperation.builder()
    .checksumAlgorithm(ComputeObjectChecksumAlgorithm.CRC64)
    .checksumType(ComputeObjectChecksumType.COMPOSITE)
    .build();

JobOperation operation = JobOperation.builder()
    .s3ComputeObjectChecksum(s3ComputeObjectChecksum)
    .build();

// Job Manifest
JobManifestLocation location = JobManifestLocation.builder()
    .eTag(manifestEtag)
    .objectArn(manifestArn)
    .build();
```

```
JobManifestSpec spec = JobManifestSpec.builder()
    .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
    .fields(Arrays.asList(JobManifestFieldName.BUCKET, JobManifestFieldName.KEY))
    .build();

JobManifest manifest = JobManifest.builder()
    .location(location)
    .spec(spec)
    .build();

// Completion Report
JobReport report = JobReport.builder()
    .bucket(reportBucketArn)
    .enabled(true) // Must be true
    .expectedBucketOwner(reportExpectedBucketOwner)
    .format(JobReportFormat.REPORT_CSV_20180820)
    .prefix(reportPrefix)
    .reportScope(JobReportScope.ALL_TASKS)
    .build();

// Create Job Request
CreateJobRequest request = CreateJobRequest.builder()
    .accountId(accountId)
    .confirmationRequired(false)
    .manifest(manifest)
    .operation(operation)
    .priority(10)
    .report(report)
    .roleArn(roleArn);

// Create the client
S3ControlClient client = S3ControlClient.builder()
    .credentialsProvider(new ProfileCredentialsProvider())
    .region(Region.US_EAST_1)
    .build();

// Send the request
try {
    CreateJobResponse response = client.createJob(request);
    System.out.println(response);
} catch (AwsServiceException e) {
    System.out.println("AwsServiceException: " + e.getMessage());
    throw new RuntimeException(e);
```

```
} catch (SdkClientException e) {
    System.out.println("SdkClientException: " + e.getMessage());
    throw new RuntimeException(e);
}
```

Example: Viewing Compute checksum job details

The following example shows how you can specify a job ID to view the job details (such as the job completion rate) for a **Compute checksum** job request:

```
DescribeJobRequest request = DescribeJobRequest.builder()
        .accountId("1234567890")
        .jobId("a16217a1-e082-48e5-b04f-31fac3a66b13")
        .build();
```

Using the AWS CLI

You can use the create-job command to create a new batch operations job, and to provide the list of objects. Then, specify the checksum algorithm and checksum type, and the destination bucket where you want to save the **Compute checksum** report. The following example creates an S3 Batch Operations **Compute checksum** job by using an S3 generated manifest for the AWS account *111122223333*.

To use this command, replace the *user input placeholders* with your own information:

```
aws s3control create-job \
    --account-id 111122223333 \
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::my-manifest-bucket/
manifest.csv","ETag":"e0e8bfc50e0f0c5d5a1a5f0e0e8bfc50"}}' \
    --manifest-generator '{
        "S3JobManifestGenerator": {
          "ExpectedBucketOwner": "111122223333",
          "SourceBucket": "arn:aws:s3:::amzn-s3-demo-source-bucket",
          "EnableManifestOutput": true,
          "ManifestOutputLocation": {
            "ExpectedManifestBucketOwner": "111122223333",
            "Bucket": "arn:aws:s3:::amzn-s3-demo-manifest-bucket",
            "ManifestPrefix": "prefix",
            "ManifestFormat": "S3InventoryReport_CSV_20211130"
          },
```

```
          "Filter": {
            "CreatedAfter": "2023-09-01",
            "CreatedBefore": "2023-10-01",
            "KeyNameConstraint": {
              "MatchAnyPrefix": [
                "prefix"
              ],
              "MatchAnySuffix": [
                "suffix"
              ]
            },
            "ObjectSizeGreaterThanBytes": 100,
            "ObjectSizeLessThanBytes": 200,
            "MatchAnyStorageClass": [
              "STANDARD",
              "STANDARD_IA"
            ]
          }
        }
      }' \
    --operation '{"S3ComputeObjectChecksum":
{"ChecksumAlgorithm":"CRC64NVME","ChecksumType":"FULL_OBJECT"}}' \
    --report '{"Bucket":"arn:aws:s3:::my-report-
bucket","Format":"Report_CSV_20180820","Enabled":true,"Prefix":"batch-op-
reports/","ReportScope":"AllTasks","ExpectedBucketOwner":"111122223333"}' \
    --priority 10 \
    --role-arn arn:aws:iam::123456789012:role/S3BatchJobRole \
    --client-request-token 6e023a7e-4820-4654-8c81-7247361aeb73 \
    --description "Compute object checksums" \
    --region us-west-2
```

After you submit the **Compute checksum** job, you receive the job ID as a response and it appears on the S3 Batch Operations list page. Amazon S3 processes the list of objects and calculates checksums for each object. After the job finishes, S3 provides a consolidated **Compute checksum** report at the specified destination.

To monitor the progress of your **Compute checksum** job, use the describe-job command. This command checks the status of the specified batch operations job. To use this command, replace the *user input placeholders* with your own information.

For example:

```
aws s3control describe-job --account-id 111122223333 --job-id 1234567890abcdef0
```

To obtain a list of all **Active** and **Complete** batch operations jobs, see Listing jobs or list-jobs in the *AWS CLI Command Reference*.

Using the REST API

You can send REST requests to verify object integrity with **Compute checksum** using CreateJob. You can monitor the progress of **Compute checksum** requests by sending REST requests to the DescribeJob API operation. Each batch operations job progresses through the following statuses:

- **NEW**
- **PREPARING**
- **READY**
- **ACTIVE**
- **PAUSING**
- **PAUSED**
- **COMPLETE**
- **CANCELLING**
- **FAILED**

The API response notifies you of the current job state.

Deleting Amazon S3 objects

You can delete one or more objects directly from Amazon S3 using the Amazon S3 console, AWS SDKs, AWS Command Line Interface (AWS CLI), or REST API. For example, if you're collecting log files, it's a good idea to delete them when they're no longer needed. You can set up an S3 Lifecycle rule to automatically delete objects such as log files.

To delete an object, you can use one of the following API operations:

- **Delete a single object** – Amazon S3 provides the `DELETE` (`DeleteObject`) API operation that you can use to delete one object in a single HTTP request.
- **Delete multiple objects** – Amazon S3 provides the Multi-Object Delete (`DeleteObjects`) API operation that you can use to delete up to 1,000 objects in a single HTTP request.

When deleting objects from a bucket that is not versioning-enabled, you provide only the object key name. However, when deleting objects from a versioning-enabled bucket, you can provide the version ID of the object to delete a specific version of the object.

Best practices to consider before deleting an object

Before you delete an object, consider the following best practices:

- Enable bucket versioning. S3 Versioning adds protection against simple `DeleteObject` requests to prevent accidental deletions. For versioned buckets, if you delete the current version of an object or when a delete request doesn't specify a specific version Id, Amazon S3 doesn't permanently delete the object. Instead, S3 adds a delete marker, issuing a soft delete of the object. The delete marker then becomes the current (or newest) version of the object with a new version ID. For more information, see Deleting object versions from a versioning-enabled bucket.
- If you want to delete a large number of objects, or for programmatically deleting objects based on object creation date, set a S3 Lifecycle configuration on your bucket. To monitor these deletions, we recommend that you use an S3 Lifecycle event notification. When you configure S3 Lifecycle notifications, the `s3:LifecycleExpiration:Delete` event type notifies you when an object in a bucket is deleted. It also notifies you when an object version is permanently deleted by an S3 Lifecycle configuration. The `s3:LifecycleExpiration:DeleteMarkerCreated` event type notifies you when S3 Lifecycle creates a delete marker. A delete marker is created when a current version of an object in a versioned bucket is deleted.
- Before making any updates to your S3 Lifecycle configuration, confirm that Lifecycle has completed the actions on all intended objects. For more information, see the **Updating, disabling, or deleting Lifecycle rules** section in Setting an S3 Lifecycle configuration on a bucket.

  > **Note**
  >
  > The S3 Lifecycle rules must apply to the right subset of objects to prevent unintended deletions. You can filter objects by prefixes, object tags, or object sizes when creating the Lifecycle rules.

- Consider restricting users from removing or deleting objects from your bucket. To restrict users, you'll need to explicitly deny users the permissions for the following actions in your Amazon S3 bucket policies:

  - `s3:DeleteObject`, `s3:DeleteObjectVersion` (to control who can delete objects using API requests)
  - `s3:PutLifecycleConfiguration` (to control who can add S3 Lifecycle expiration rules)
- Consider using S3 Replication to create multiple copies of your data and to replicate them to multiple locations at once. You can choose as many destination buckets as needed. Additionally, if an object is unintentionally deleted, you'll still have a copy of the data.
- Avoid sending object version IDs to unversioned buckets. Also, make sure to properly set both the `s3:DeleteObject` and `s3:DeleteObjectVersion` permissions on all buckets (including unversioned buckets).

Deleting objects from a versioning-enabled bucket

If your bucket is versioning-enabled, multiple versions of the same object can exist in the bucket. When working with versioning-enabled buckets, the `Delete` API operations enable the following options:

- **Specify a non-versioned delete request** – Specify only the object's key, and not the version ID. In this case, Amazon S3 creates a delete marker over the current version of the object and returns its version ID in the response. This makes your object disappear from the bucket. For information about object versioning and the delete marker concept, see Retaining multiple versions of objects with S3 Versioning.
- **Specify a versioned delete request** – Specify both the key and version ID. In this case, the following outcomes are possible:
  - If the version ID maps to a specific object version, Amazon S3 deletes the specific version of the object.
  - If the version ID maps to the delete marker of an object, Amazon S3 deletes the delete marker. When the delete marker gets deleted, the object then reappears in your bucket.

Deleting objects from a versioning-suspended bucket

If your bucket is versioning-suspended, the `Delete` API operations behave the same way for versioning enabled buckets (except for when the current version has a null version ID). For more information, see Deleting objects from versioning-suspended buckets.

Deleting objects from an unversioned bucket

If your bucket is unversioned, you can specify the object's key in the `Delete` API operations and Amazon S3 will permanently delete the object. To prevent permanent deletion of an object, enable bucket versioning.

For unversioned buckets, if the `s3:DeleteObject` or `s3:DeleteObjectVersion` permissions are explicitly denied in your bucket policy, then any `DeleteObject`, `DeleteObjects`, or `DeleteObjectVersion` requests result in a `403 Access Denied` error.

Deleting objects from an MFA-enabled bucket

When deleting objects from a multi-factor authentication (MFA)-enabled bucket, note the following:

- If you provide an MFA token that isn't valid, the request always fails.
- If you have an MFA-enabled bucket and you make a versioned delete request (you provide an object key and version ID), the request fails if you don't provide a valid MFA token. In addition, when using the multi-object `Delete` API operation on an MFA-enabled bucket, if any of the deletes are a versioned delete request (that is, you specify an object key and version ID), the entire request fails if you don't provide an MFA token.

However, in the following cases, the request succeeds:

- If you have an MFA-enabled bucket and you make a non-versioned delete request (you are not deleting a versioned object), and you don't provide an MFA token, the delete succeeds.
- If you have a Multi-Object Delete request that specifies only non-versioned objects to delete from an MFA-enabled bucket and you don't provide an MFA token, the deletions succeed.

For information about MFA delete, see Configuring MFA delete.

Topics

- Deleting a single object
- Deleting multiple objects

Deleting a single object

You can use the Amazon S3 console or the DELETE API to delete a single existing object from an S3 bucket. For more information about deleting objects in Amazon S3, see Deleting Amazon S3 objects.

Because all objects in your S3 bucket incur storage costs, you should delete objects that you no longer need. For example, if you are collecting log files, it's a good idea to delete them when they're no longer needed. You can set up a lifecycle rule to automatically delete objects such as log files. For more information, see the section called “Setting lifecycle configuration”.

For information about Amazon S3 features and pricing, see Amazon S3 pricing.

Using the S3 console

Follow these steps to use the Amazon S3 console to delete a single object from a bucket.

> **Warning**
>
> When you permanently delete an object or specified object version in the Amazon S3 console, the deletion can't be undone.

To delete an object that has versioning enabled or suspended

> **Note**
>
> If the version ID for an object in a versioning-suspended bucket is marked as NULL, S3 permanently deletes the object since no previous versions exist. However, if a valid version ID is listed for the object in a versioning-suspended bucket, then S3 creates a delete marker for the deleted object, while retaining the previous versions of the object.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets** or **Directory buckets**.
3. In the bucket list, choose the name of the bucket that you want to delete an object from.
4. Select the object and then choose **Delete**.

5. To confirm deletion of the objects list under **Specified objects** in the **Delete objects?** text box, enter **`delete`**.

To permanently delete a specific object version in a versioning-enabled bucket

> **Warning**
>
> When you permanently delete a specific object version in Amazon S3, the deletion can't be undone.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the name of the bucket that you want to delete an object from.
3. Select the object that you want to delete.
4. Choose the **Show versions** toggle.
5. Select the object version and then choose **Delete**.
6. To confirm permanent deletion of the specific object versions listed under **Specified objects**, in the **Delete objects?** text box, enter **Permanently delete**. Amazon S3 permanently deletes the specific object version.

To permanently delete an object in an Amazon S3 bucket that *doesn't* have versioning enabled

> **Warning**
>
> When you permanently delete an object in Amazon S3, the deletion can't be undone. Also, for any buckets without versioning enabled, deletions are permanent.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the bucket list, choose the name of the bucket that you want to delete an object from.
4. Select the object and then choose **Delete**.

5. To confirm permanent deletion of the object listed under **Specified objects**, in the **Delete objects?** text box, enter **permanently delete**.

> **Note**
>
> If you're experiencing any issues with deleting your object, see I want to permanently delete versioned objects.

Using the AWS CLI

To delete one object per request, use the DELETE API. For more information, see DELETE Object. For more information about using the CLI to delete an object, see delete-object.

Using the REST API

You can use the AWS SDKs to delete an object. However, if your application requires it, you can send REST requests directly. For more information, see DELETE Object in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

The following examples show how you can use the AWS SDKs to delete an object from a bucket. For more information, see DELETE Object in the *Amazon Simple Storage Service API Reference*

If you have S3 Versioning enabled on the bucket, you have the following options:

- Delete a specific object version by specifying a version ID.
- Delete an object without specifying a version ID, in which case Amazon S3 adds a delete marker to the object.

For more information about S3 Versioning, see Retaining multiple versions of objects with S3 Versioning.

For more examples, and examples in other languages, see Use `DeleteObject` with an AWS SDK or CLI in the *Amazon S3 API reference*.

Java

To delete objects using the AWS SDK for Java, you can delete objects from both versioned and non-versioned buckets:

- *Non-versioned bucket:* In the delete request, you specify only the object key and not a version ID.
- *Versioned bucket:* You can delete a specific object version by specifying both the object key name and version ID. If there are no other versions of that object, Amazon S3 deletes the object entirely. Otherwise, Amazon S3 only deletes the specified version.

> **Note**
>
> You can get the version IDs of an object by sending a `ListVersions` request.

For examples of how to delete a single object with the AWS SDK for Java, see Delete an object in the *Amazon S3 API Reference*.

.NET

The following examples show how to delete an object from both versioned and non-versioned buckets. For more information about S3 Versioning, see Retaining multiple versions of objects with S3 Versioning.

Example Deleting an object from a non-versioned bucket

The following C# example deletes an object from a non-versioned bucket. The example assumes that the objects don't have version IDs, so you don't specify version IDs. You specify only the object key.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
```

```
{
    class DeleteObjectNonVersionedBucketTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            DeleteObjectNonVersionedBucketAsync().Wait();
        }
        private static async Task DeleteObjectNonVersionedBucketAsync()
        {
            try
            {
                var deleteObjectRequest = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(deleteObjectRequest);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 deleting an object", e.Message);
            }
        }
    }
}
```

Example Deleting an object from a versioned bucket

The following C# example deletes an object from a versioned bucket. It deletes a specific version of the object by specifying the object key name and version ID.

The code performs the following tasks:

1. Enables S3 Versioning on a bucket that you specify (if S3 Versioning is already enabled, this has no effect).
2. Adds a sample object to the bucket. In response, Amazon S3 returns the version ID of the newly added object. The example uses this version ID in the delete request.
3. Deletes the sample object by specifying both the object key name and a version ID.

> **Note**
>
> You can also get the version ID of an object by sending a `ListVersions` request.
>
> ```
> var listResponse = client.ListVersions(new ListVersionsRequest { BucketName
>  = bucketName, Prefix = keyName });
> ```

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DeleteObjectVersion
    {
        private const string bucketName = "*** versioning-enabled bucket name ***";
        private const string keyName = "*** Object Key Name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
```

```
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateAndDeleteObjectVersionAsync().Wait();
        }

        private static async Task CreateAndDeleteObjectVersionAsync()
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(keyName);

                // Delete the object by specifying an object key and a version ID.
                DeleteObjectRequest request = new DeleteObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID
                };
                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
deleting an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
deleting an object", e.Message);
            }
        }

        static async Task<string> PutAnObject(string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                ContentBody = "This is the content body!"
            };
```

```
            PutObjectResponse response = await client.PutObjectAsync(request);
            return response.VersionId;
        }
    }
}
```

PHP

This example shows how to use classes from version 3 of the AWS SDK for PHP to delete an object from a non-versioned bucket. For information about deleting an object from a versioned bucket, see Using the REST API.

For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

The following PHP example deletes an object from a bucket. Because this example shows how to delete objects from non-versioned buckets, it provides only the bucket name and object key (not a version ID) in the delete request.

```
<?php

require 'vendor/autoload.php';

use Aws\S3\S3Client;
use Aws\S3\Exception\S3Exception;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// 1. Delete the object from the bucket.
try
{
    echo 'Attempting to delete ' . $keyname . '...' . PHP_EOL;

    $result = $s3->deleteObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);
```

```
    if ($result['DeleteMarker'])
    {
        echo $keyname . ' was deleted or does not exist.' . PHP_EOL;
    } else {
        exit('Error: ' . $keyname . ' was not deleted.' . PHP_EOL);
    }
}
catch (S3Exception $e) {
    exit('Error: ' . $e->getAwsErrorMessage() . PHP_EOL);
}

// 2. Check to see if the object was deleted.
try
{
    echo 'Checking to see if ' . $keyname . ' still exists...' . PHP_EOL;

    $result = $s3->getObject([
        'Bucket' => $bucket,
        'Key'    => $keyname
    ]);

    echo 'Error: ' . $keyname . ' still exists.';
}
catch (S3Exception $e) {
    exit($e->getAwsErrorMessage());
}
```

Javascript

```
import { DeleteObjectCommand } from "@aws-sdk/client-s3";
import { s3Client } from "./libs/s3Client.js" // Helper function that creates Amazon
 S3 service client module.

export const bucketParams = { Bucket: "BUCKET_NAME", Key: "KEY" };

export const run = async () => {
  try {
    const data = await s3Client.send(new DeleteObjectCommand(bucketParams));
    console.log("Success. Object deleted.", data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err);
  }
```

```
};
run();
```

Deleting multiple objects

Because all objects in your S3 bucket incur storage costs, you should delete objects that you no longer need. For example, if you are collecting log files, it's a good idea to delete them when they're no longer needed. You can set up a lifecycle rule to automatically delete objects such as log files. For more information, see the section called “Setting lifecycle configuration”.

For information about Amazon S3 features and pricing, see Amazon S3 pricing.

You can use the Amazon S3 console, AWS SDKs, or the REST API to delete multiple objects simultaneously from an S3 bucket.

Using the S3 console

Follow these steps to use the Amazon S3 console to delete multiple objects from a bucket.

Warning

- Deleting a specified object cannot be undone.
- This action deletes all specified objects. When deleting folders, wait for the delete action to finish before adding new objects to the folder. Otherwise, new objects might be deleted as well.
- When deleting objects in a bucket without versioning enabled, including directory buckets, Amazon S3 will permanently delete the objects.
- When deleting objects in a bucket with bucket versioning **enabled** or **suspended**, Amazon S3 creates delete markers. For more information, see Working with delete markers.

To delete objects that have versioning enabled or suspended

Note

If the version IDs for the object in a versioning-suspended bucket are marked as NULL, S3 permanently deletes the objects since no previous versions exist. However, if a valid version

ID is listed for the objects in a versioning-suspended bucket, then S3 creates delete markers for the deleted objects, while retaining the previous versions of the objects.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the bucket list, choose the name of the bucket that you want to delete the objects from.
4. Select the objects and then choose **Delete**.
5. To confirm deletion of the objects list under **Specified objects** in the **Delete objects?** text box, enter **`delete`**.

To permanently delete specific object versions in a versioning-enabled bucket

Warning

When you permanently delete specific object versions in Amazon S3, the deletion can't be undone.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the bucket list, choose the name of the bucket that you want to delete the objects from.
4. Select the objects that you want to delete.
5. Choose the **Show versions** toggle.
6. Select the object versions and then choose **Delete**.
7. To confirm permanent deletion of the specific object versions listed under **Specified objects**, in the **Delete objects?** text box, enter **Permanently delete**. Amazon S3 permanently deletes the specific object versions.

To permanently delete the objects in an Amazon S3 bucket that *don't* have versioning enabled

> **Warning**
>
> When you permanently delete an object in Amazon S3, the deletion can't be undone. Also, for any buckets without versioning enabled, including directory buckets, deletions are permanent.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets** or **Directory buckets**.
3. In the bucket list, choose the name of the bucket that you want to delete the objects from.
4. Select the objects and then choose **Delete**.
5. To confirm permanent deletion of the objects listed under **Specified objects**, in the **Delete objects?** text box, enter **permanently delete**.

> **Note**
>
> If you're experiencing any issues with deleting your objects, see I want to permanently delete versioned objects.

Using the AWS SDKs

For examples of how to delete multiple objects with the AWS SDKs, see Delete multiple objects in the *AWS SDK Code Examples*.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *AWS SDK Code Examples*.

Using the REST API

You can use the AWS SDKs to delete multiple objects using the Multi-Object Delete API. However, if your application requires it, you can send REST requests directly.

For more information, see Delete Multiple Objects in the *Amazon Simple Storage Service API Reference*.

Organizing, listing, and working with your objects

In Amazon S3, you can use prefixes to organize your storage. A prefix is a logical grouping of the objects in a bucket. The prefix value is similar to a directory name that enables you to store similar data under the same directory in a bucket. When you programmatically upload objects, you can use prefixes to organize your data.

In the Amazon S3 console, prefixes are called folders. You can view all your objects and folders in the S3 console by navigating to a bucket. You can also view information about each object, including object properties.

For more information about listing and organizing your data in Amazon S3, see the following topics.

Topics

- Organizing objects using prefixes
- Listing object keys programmatically
- Organizing objects in the Amazon S3 console by using folders
- Viewing object properties in the Amazon S3 console

Organizing objects using prefixes

You can use prefixes to organize the data that you store in Amazon S3 buckets. A prefix is a string of characters at the beginning of the object key name. A prefix can be any length, subject to the maximum length of the object key name (1,024 bytes). You can think of prefixes as a way to organize your data in a similar way to directories. However, prefixes are not directories.

Searching by prefix limits the results to only those keys that begin with the specified prefix. The delimiter causes a list operation to roll up all the keys that share a common prefix into a single summary list result.

The purpose of the prefix and delimiter parameters is to help you organize and then browse your keys hierarchically. To do this, first pick a delimiter for your bucket, such as slash (/), that doesn't occur in any of your anticipated key names. You can use another character as a delimiter. There is nothing unique about the slash (/) character, but it is a very common prefix delimiter. Next, construct your key names by concatenating all containing levels of the hierarchy, separating each level with the delimiter.

For example, if you were storing information about cities, you might naturally organize them by continent, then by country, then by province or state. Because these names don't usually contain punctuation, you might use slash (/) as the delimiter. The following examples use a slash (/) delimiter.

- Europe/France/Nouvelle-Aquitaine/Bordeaux
- North America/Canada/Quebec/Montreal
- North America/USA/Washington/Bellevue
- North America/USA/Washington/Seattle

If you stored data for every city in the world in this manner, it would become awkward to manage a flat key namespace. By using `Prefix` and `Delimiter` with the list operation, you can use the hierarchy that you've created to list your data. For example, to list all the states in USA, set `Delimiter='/'` and `Prefix='North America/USA/'`. To list all the provinces in Canada for which you have data, set `Delimiter='/'` and `Prefix='North America/Canada/'`.

For more information about delimiters, prefixes, and nested folders, see Difference between prefixes and nested folders.

Listing objects using prefixes and delimiters

If you issue a list request with a delimiter, you can browse your hierarchy at only one level, skipping over and summarizing the (possibly millions of) keys nested at deeper levels. For example, assume that you have a bucket (*amzn-s3-demo-bucket*) with the following keys:

```
sample.jpg
```

```
photos/2006/January/sample.jpg
```

```
photos/2006/February/sample2.jpg
```

```
photos/2006/February/sample3.jpg
```

```
photos/2006/February/sample4.jpg
```

The sample bucket has only the `sample.jpg` object at the root level. To list only the root level objects in the bucket, you send a GET request on the bucket with the slash (/) delimiter character. In response, Amazon S3 returns the `sample.jpg` object key because it does not contain the / delimiter character. All other keys contain the delimiter character. Amazon S3 groups these keys

and returns a single `CommonPrefixes` element with the prefix value `photos/`, which is a substring from the beginning of these keys to the first occurrence of the specified delimiter.

Example

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>amzn-s3-demo-bucket</Name>
  <Prefix></Prefix>
  <Marker></Marker>
  <MaxKeys>1000</MaxKeys>
  <Delimiter>/</Delimiter>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>sample.jpg</Key>
    <LastModified>2011-07-24T19:39:30.000Z</LastModified>
    <ETag>&quot;d1a7fb5eab1c16cb4f7cf341cf188c3d&quot;</ETag>
    <Size>6</Size>
    <Owner>
      <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <CommonPrefixes>
    <Prefix>photos/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

For more information about listing object keys programmatically, see Listing object keys programmatically.

Listing object keys programmatically

In Amazon S3, keys can be listed by prefix. You can choose a common prefix for the names of related keys and mark these keys with a special character that delimits hierarchy. You can then use the list operation to select and browse keys hierarchically. This is similar to how files are stored in directories within a file system.

Amazon S3 exposes a list operation that lets you enumerate the keys contained in a bucket. Keys are selected for listing by bucket and prefix. For example, consider a bucket named "`dictionary`" that contains a key for every English word. You might make a call to list all the keys in that bucket that start with the letter "q". List results are always returned in UTF-8 binary order.

Both the SOAP and REST list operations return an XML document that contains the names of matching keys and information about the object identified by each key.

> **Note**
>
> SOAP APIs for Amazon S3 are not available for new customers, and are approaching End of Life (EOL) on August 31, 2025. We recommend that you use either the REST API or the AWS SDKs.

Groups of keys that share a prefix terminated by a special delimiter can be rolled up by that common prefix for the purposes of listing. This enables applications to organize and browse their keys hierarchically, much like how you would organize your files into directories in a file system.

For example, to extend the dictionary bucket to contain more than just English words, you might form keys by prefixing each word with its language and a delimiter, such as `"French/logical"`. Using this naming scheme and the hierarchical listing feature, you could retrieve a list of only French words. You could also browse the top-level list of available languages without having to iterate through all the lexicographically intervening keys. For more information about this aspect of listing, see Organizing objects using prefixes.

REST API

If your application requires it, you can send REST requests directly. You can send a GET request to return some or all of the objects in a bucket or you can use selection criteria to return a subset of the objects in a bucket. For more information, see GET Bucket (List Objects) Version 2 in the *Amazon Simple Storage Service API Reference*.

List implementation efficiency

List performance is not substantially affected by the total number of keys in your bucket. It's also not affected by the presence or absence of the `prefix`, `marker`, `maxkeys`, or `delimiter` arguments.

Iterating through multipage results

As buckets can contain a virtually unlimited number of keys, the complete results of a list query can be extremely large. To manage large result sets, the Amazon S3 API supports pagination to split them into multiple responses. Each list keys response returns a page of up to 1,000 keys with

an indicator indicating if the response is truncated. You send a series of list keys requests until you have received all the keys. AWS SDK wrapper libraries provide the same pagination.

Examples

When you list all of the objects in your bucket, note that you must have the `s3:ListBucket` permission.

CLI

list-objects

The following example uses the `list-objects` command to display the names of all the objects in the specified bucket:

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

The example uses the `--query` argument to filter the output of `list-objects` down to the key value and size for each object

For more information about objects, see Working with objects in Amazon S3.

- For API details, see ListObjects in *AWS CLI Command Reference*.

ls

The following example lists all objects and prefixes in a bucket by using the `ls` command.

To use this example command, replace *amzn-s3-demo-bucket* with the name of your bucket.

```
$ aws s3 ls s3://amzn-s3-demo-bucket
```

- For more information about the high-level command `ls`, see List buckets and objects in *AWS Command Line Interface User Guide*.

PowerShell

Tools for PowerShell V4

Example 1: This command retrieves the information about all of the items in the bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Example 2: This command retrieves the information about the item "sample.txt" from bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Example 3: This command retrieves the information about all items with the prefix "sample" from bucket "test-files".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- For API details, see ListObjects in *AWS Tools for PowerShell Cmdlet Reference (V4)*.

Tools for PowerShell V5

Example 1: This command retrieves the information about all of the items in the bucket "amzn-s3-demo-bucket".

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Example 2: This command retrieves the information about the item "sample.txt" from bucket "amzn-s3-demo-bucket".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Example 3: This command retrieves the information about all items with the prefix "sample" from bucket "amzn-s3-demo-bucket".

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- For API details, see ListObjects in *AWS Tools for PowerShell Cmdlet Reference (V5)*.

Organizing objects in the Amazon S3 console by using folders

In Amazon S3 general purpose buckets, objects are the primary resources, and objects are stored in buckets. Amazon S3 general purpose buckets have a flat structure instead of a hierarchy like you would see in a file system. However, for the sake of organizational simplicity, the Amazon S3

console supports the *folder* concept as a means of grouping objects. The console does this by using a shared name *prefix* for the grouped objects. In other words, the grouped objects have names that begin with a common string. This common string, or shared prefix, is the folder name. Object names are also referred to as *key names*.

For example, you can create a folder in a general purpose bucket in the console named `photos` and store an object named `myphoto.jpg` in it. The object is then stored with the key name `photos/myphoto.jpg`, where `photos/` is the prefix.

Here are two more examples:

- If you have three objects in your general purpose bucket—`logs/date1.txt`, `logs/date2.txt`, and `logs/date3.txt`—the console will show a folder named `logs`. If you open the folder in the console, you will see three objects: `date1.txt`, `date2.txt`, and `date3.txt`.
- If you have an object named `photos/2017/example.jpg`, the console shows you a folder named `photos` that contains the folder `2017`. The folder `2017` contains the object `example.jpg`.

You can have folders within folders, but not buckets within buckets. You can upload and copy objects directly into a folder. Folders can be created, deleted, and made public, but they can't be renamed. Objects can be copied from one folder to another.

Important

When you create a folder in Amazon S3 console, S3 creates a 0-byte object. This object key is set to the folder name that you provided plus a trailing forward slash (/) character. For example, in Amazon S3 console, if you create a folder named `photos` in your bucket, the Amazon S3 console creates a 0-byte object with the key `photos/`. The console creates this object to support the idea of folders.

Also, any pre-existing object that's named with a trailing forward slash character (/) appears as a folder in the Amazon S3 console. For example, an object with the key name `examplekeyname/` appears as a folder in Amazon S3 console and not as an object. Otherwise, it behaves like any other object and can be viewed and manipulated through the AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. Additionally, you can't upload an object that has a key name with a trailing forward slash character (/) by using the Amazon S3 console. However, you can upload objects that are named with a trailing forward slash (/) character by using the AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API.

Moreover, the Amazon S3 console doesn't display the content and metadata for folder objects like it does for other objects. When you use the console to copy an object named with a trailing forward slash character (/), a new folder is created in the destination location, but the object's data and metadata aren't copied. Also, a forward slash (/) in object key names might require special handling. For more information, see Naming Amazon S3 objects.

To create folders in directory buckets, upload a folder. For more information, see Uploading objects to a directory bucket.

Topics

- Creating a folder
- Making folders public
- Calculating folder size
- Deleting folders

Creating a folder

This section describes how to use the Amazon S3 console to create a folder.

Important

If your bucket policy prevents uploading objects to this bucket without tags, metadata, or access control list (ACL) grantees, you can't create a folder by using the following procedure. Instead, upload an empty folder and specify the following settings in the upload configuration.

To create a folder

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to create a folder in.
4. On the **Objects** tab, choose **Create folder**.

5. Enter a name for the folder (for example, **`favorite-pics`**).

> **Note**
>
> Folder names are subject to certain limitations and guidelines, and are considered part of an object's object key name, which is limited to 1,024 bytes. For more information, see the section called “Naming objects”.

6. (Optional) If your bucket policy requires objects to be encrypted with a specific encryption key, under **Server-side encryption**, you must choose **Specify an encryption key** and specify the same encryption key when you create a folder. Otherwise, folder creation will fail.
7. Choose **Create folder**.

Making folders public

We recommend blocking all public access to your Amazon S3 folders and buckets unless you specifically require a public folder or bucket. When you make a folder public, anyone on the internet can view all the objects that are grouped in that folder.

In the Amazon S3 console, you can make a folder public. You can also make a folder public by creating a bucket policy that limits data access by prefix. For more information, see Identity and Access Management for Amazon S3.

> **Warning**
>
> After you make a folder public in the Amazon S3 console, you can't make it private again. Instead, you must set permissions on each individual object in the public folder so that the objects have no public access. For more information, see Configuring ACLs.

Topics

- Calculating folder size
- Deleting folders

Calculating folder size

This section describes how to use the Amazon S3 console to calculate a folder's size.

To calculate a folder's size

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the **General purpose buckets** list, choose the name of the bucket in which your folder is stored.
4. In the **Objects** list, select the checkbox next to the name of the folder.
5. Choose **Actions**, and then choose **Calculate total size**.

Note

When you navigate away from the page, the folder information (including the total size) is no longer available. You must calculate the total size again if you want to see it again.

Important

When you use the **Calculate total size** action on specified objects or folders within your bucket, Amazon S3 calculates the total number of objects and the total storage size. However, incomplete or in-progress multipart uploads and previous or noncurrent versions aren't calculated in the total number of objects or the total size. This action calculates only the total number of objects and the total size for the current or newest version of each object that's stored in the bucket.

For example, if there are two versions of an object in your bucket, then the storage calculator in Amazon S3 counts them as only one object. As a result, the total number of objects that's calculated in the Amazon S3 console can differ from the **Object Count** metric shown in S3 Storage Lens and from the number reported by the Amazon CloudWatch metric, `NumberOfObjects`. Likewise, the total storage size can also differ from the **Total Storage** metric shown in S3 Storage Lens and from the `BucketSizeBytes` metric shown in CloudWatch.

Deleting folders

Use the Amazon S3 console to delete folders from an S3 bucket.

For information about Amazon S3 features and pricing, see Amazon S3.

To delete folders from an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the **General purpose buckets** list, choose the name of the bucket that you want to delete folders from.
4. In the **Objects** list, select the checkboxes next to the folders and objects that you want to delete.
5. Choose **Delete**.
6. On the **Delete objects** page, verify that the folders and objects you chose are listed under **Specified objects**.
7. In the **Delete objects** box, enter **`delete`**, and choose **Delete objects**.

> ⚠ **Warning**
>
> This action deletes all specified objects. When you delete folders, wait for the delete to finish before you add new objects to the folder. Otherwise, new objects might be deleted as well.

Viewing object properties in the Amazon S3 console

You can use the Amazon S3 console to view the properties of an object, including storage class, encryption settings, tags, and metadata.

To view the properties of an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets** or **Directory buckets**.
3. In the bucket list, choose the name of the bucket that contains the object.
4. In the **Objects** list, choose the name of the object you want to view properties for.

The **Object overview** for your object opens. You can scroll down to view the object properties.

5. On the **Object overview** page, you can view or configure the following properties for the object.

> **Note**
>
> - If you change the **Storage Class**, **Encryption**, or **Metadata** properties, a new object is created to replace the old one. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. The role that changes the property also becomes the owner of the new object or (object version).
> - If you change the **Storage Class**, **Encryption**, or **Metadata** properties for an object that has user-defined tags, you must have the `s3:GetObjectTagging` permission. If you're changing these properties for an object that doesn't have user-defined tags but is over 16 MB in size, you must also have the `s3:GetObjectTagging` permission.
>
> If the destination bucket policy denies the `s3:GetObjectTagging` action, these properties for the object will be updated, but the user-defined tags will be removed from the object, and you will receive an error.

a. **Storage class** – Each object in Amazon S3 has a storage class associated with it. The storage class that you choose to use depends on how frequently you access the object. The default storage class for S3 objects in general purpose buckets is STANDARD. The default storage class for S3 objects in directory buckets is S3 Express One Zone. You choose which storage class to use when you upload an object. For more information about storage classes, see Understanding and managing Amazon S3 storage classes.

   To change the storage class after you upload an object to a general purpose bucket, choose **Storage class**. Choose the storage class that you want, and then choose **Save**.

   > **Note**
   >
   > Storage class of objects in a directory bucket cannot be changed.

b. **Server-side encryption settings** – You can use server-side encryption to encrypt your S3 objects. For more information, see Specifying server-side encryption with AWS KMS (SSE-KMS) or Specifying server-side encryption with Amazon S3 managed keys (SSE-S3).

c. **Metadata** – Each object in Amazon S3 has a set of name-value pairs that represents its metadata. For information about adding metadata to an S3 object, see Editing object metadata in the Amazon S3 console.

d. **Tags** – You categorize storage by adding tags to an S3 object in a general purpose bucket. For more information, see Tagging your objects.

e. **Object lock legal hold and retention** – You can prevent an object in a general purpose bucket from being deleted. For more information, see Locking objects with Object Lock.

Download and upload objects with presigned URLs

You can use presigned URLs to grant time-limited access to objects in Amazon S3 without updating your bucket policy. A presigned URL can be entered in a browser or used by a program to download an object. The credentials used by the presigned URL are those of the AWS Identity and Access Management (IAM) principal who generated the URL.

You can also use presigned URLs to allow someone to upload a specific object to your Amazon S3 bucket. This allows an upload without requiring another party to have AWS security credentials or permissions. If an object with the same key already exists in the bucket as specified in the presigned URL, Amazon S3 replaces the existing object with the uploaded object.

You can use the presigned URL multiple times, up to the expiration date and time.

Annotation API operations (`PutObjectAnnotation`, `GetObjectAnnotation`, `ListObjectAnnotations`, and `DeleteObjectAnnotation`) support presigned URLs.

When you create a presigned URL, you must provide your security credentials, and then specify the following:

- An Amazon S3 bucket
- An object key (if downloading this object will be in your Amazon S3 bucket, if uploading this is the file name to be uploaded)
- An HTTP method (`GET` for downloading objects, `PUT` for uploading, `HEAD` for reading object metadata, etc)

- An expiration time interval

When using presigned URLs to upload objects, you can verify object integrity using checksums. While presigned URLs created with AWS Signature Version 2 only support MD5 checksums, presigned URLs created with AWS Signature Version 4 support additional checksum algorithms including CRC-64/NVME, CRC32, CRC32C, SHA-1, SHA-256, MD5, XXHash64, XXHash3, XXHash128, and SHA-512. To use these additional checksum algorithms, ensure you're using AWS Signature Version 4 and include the appropriate checksum header in your upload request. For more information about object integrity, see Checking object integrity in Amazon S3.

Topics

- Who can create a presigned URL
- Expiration time for presigned URLs
- Limiting presigned URL capabilities
- Frequently asked questions for presigned URLs
- Sharing objects with presigned URLs
- Uploading objects with presigned URLs

Who can create a presigned URL

Anyone with valid security credentials can create a presigned URL. But for someone to successfully access an object, the presigned URL must be created by someone who has permission to perform the operation that the presigned URL is based upon.

The following are the types of credentials that you can use to create a presigned URL:

- **IAM user** – Valid up to 7 days when you're using AWS Signature Version 4.

  To create a presigned URL that's valid for up to 7 days, first delegate IAM user credentials (the access key and secret key) to the method you're using to create the presigned URL.
- **Temporary security credentials** – Can't be valid for longer than the credentials themselves. These credentials include:
  - **IAM role credentials** – The presigned URL expires when the role session expires, even if you specify a longer expiration time.
  - **IAM role credentials used by Amazon EC2 instances** – Valid for the duration of the role credentials (typically 6 hours).

- **AWS Security Token Service credentials** – Valid only for the duration of the temporary credentials.

> **Note**
>
> If you created a presigned URL using a temporary credential, the URL expires when the credential expires. In general, a presigned URL expires when the credential you used to create it is revoked, deleted, or deactivated. This is true even if the URL was created with a later expiration time. For temporary security credentials lifetimes, see Comparing AWS STS API operations in the *IAM User Guide*.

Expiration time for presigned URLs

A presigned URL remains valid for the period of time specified when the URL is generated. If you create a presigned URL with the Amazon S3 console, the expiration time can be set between 1 minute and 12 hours. If you use the AWS CLI or AWS SDKs, the expiration time can be set as high as 7 days.

If you created a presigned URL by using a temporary token, then the URL expires when the token expires. In general, a presigned URL expires when the credential you used to create it is revoked, deleted, or deactivated. This is true even if the URL was created with a later expiration time. For more information about how the credentials you use affect the expiration time, see Who can create a presigned URL.

Amazon S3 checks the expiration date and time of a signed URL at the time of the HTTP request. For example, if a client begins to download a large file immediately before the expiration time, the download continues even if the expiration time passes during the download. However, if the connection drops and the client tries to restart the download after the expiration time passes, the download fails.

Limiting presigned URL capabilities

The capabilities of a presigned URL are limited by the permissions of the user who created it. In essence, presigned URLs are bearer tokens that grant access to those who possess them. As such, we recommend that you protect them appropriately. The following are some methods that you can use to restrict the use of your presigned URLs.

AWS Signature Version 4 (SigV4)

To enforce specific behavior when presigned URL requests are authenticated by using AWS Signature Version 4 (SigV4), you can use condition keys in bucket policies and access point policies. For example, the following bucket policy uses the `s3:signatureAge` condition to deny any Amazon S3 presigned URL request on objects in the *`amzn-s3-demo-bucket`* bucket if the signature is more than 10 minutes old. To use this example, replace the *`user input placeholders`* with your own information.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Deny a presigned URL request if the signature is more than 10
 min old",
            "Effect": "Deny",
            "Principal": {
                "AWS": "*"
            },
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "NumericGreaterThan": {
                    "s3:signatureAge": "600000"
                }
            }
        }
    ]
}
```

For more information about policy keys related AWS Signature Version 4, see AWS Signature Version 4 Authentication in the *Amazon Simple Storage Service API Reference*.

Network path restriction

If you want to restrict the use of presigned URLs and all Amazon S3 access to particular network paths, you can write AWS Identity and Access Management (IAM) policies. You can set these policies on the IAM principal that makes the call, the Amazon S3 bucket, or both.

A network-path restriction on the IAM principal requires the user of those credentials to make requests from the specified network. A restriction on the bucket or access point requires that all requests to that resource originate from the specified network. These restrictions also apply outside of the presigned URL scenario.

The IAM global condition key that you use depends on the type of endpoint. If you're using the public endpoint for Amazon S3, use `aws:SourceIp`. If you're using a virtual private cloud (VPC) endpoint to Amazon S3, use `aws:SourceVpc` or `aws:SourceVpce`.

The following IAM policy statement requires the principal to access AWS only from the specified network range. With this policy statement, all access must originate from that range. This includes the case of someone who's using a presigned URL for Amazon S3. To use this example, replace the *`user input placeholders`* with your own information.

```
{
    "Sid": "NetworkRestrictionForIAMPrincipal",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
        "NotIpAddressIfExists": {"aws:SourceIp": "IP-address-range"},
        "BoolIfExists": {"aws:ViaAWSService": "false"}
    }
}
```

Frequently asked questions for presigned URLs

Q: Why do my presigned URLs expire earlier than the configured expiration time?

Presigned URLs remain valid only while their underlying credentials are valid. A presigned URL expires at either its configured expiration time or when its associated credentials expire, whichever occurs first. For Amazon Elastic Container Service tasks or containers, role credentials typically rotate every 1-6 hours. When using AWS Security Token Service (AWS STS) AssumeRole, the presigned URL expires when the role session ends, which by default is 1 hour. For Amazon EC2 instance profiles, metadata credentials rotate periodically with a maximum validity period of approximately 6 hours.

Q: Why am I getting a 403 Forbidden error when accessing a presigned URL?

Before generating a presigned URL, verify that you have the correct permissions configured. The IAM user or role generating the URL must have the required permissions, such as `s3:GetObject`,

for the specific operation. Additionally, check that the Amazon S3 bucket policy doesn't explicitly deny access to the object.

Q: Why am I getting an `AccessDenied` error with `HeadersNotSigned: if-range`?

When `Range` is included in `X-Amz-SignedHeaders`, Amazon S3 requires that `If-Range` also be signed if present in the request. Add `If-Range` to `X-Amz-SignedHeaders` when you generate the presigned URL.

Q: I'm getting `SignatureDoesNotMatch` errors. How do I fix this?

If you encounter `SignatureDoesNotMatch` errors when using Amazon S3 presigned URLs, consider several common causes. First, ensure that your system clock is synchronized with a Network Time Protocol (NTP) server, as even small time drifts can invalidate signatures. Next, be aware that some corporate proxies might modify headers or query strings, potentially causing signature mismatches. To troubleshoot, try testing without the proxy. Finally, verify that all request parameters—including the HTTP method, headers, and query string—match exactly between URL generation and usage. Addressing these issues can often resolve `SignatureDoesNotMatch` errors.

Q: I'm getting `ExpiredToken` errors. What should I do?

When you receive `ExpiredToken` errors while using presigned URLs, it indicates that the AWS credentials used to generate the URL are no longer valid. To resolve this issue, refresh your AWS credentials before generating new presigned URLs. For long-running applications, we recommend implementing credential refresh logic to maintain continuous access. Where appropriate, you can use longer-lived credentials or implement token refresh mechanisms. If you're using AWS Security Token Service (AWS STS) AssumeRole, verify that your configured session duration meets your use case requirements. Remember that presigned URLs remain valid only for the duration of their underlying credentials, so implementing proper credential management is essential.

Sharing objects with presigned URLs

By default, all Amazon S3 objects are private, only the object owner has permission to access them. However, the object owner may share objects with others by creating a presigned URL. A presigned URL uses security credentials to grant time-limited permission to download objects. The URL can be entered in a browser or used by a program to download the object. The credentials used by the presigned URL are those of the AWS user who generated the URL.

For general information about presigned URLs, see Download and upload objects with presigned URLs.

You can create a presigned URL for sharing an object without writing any code by using the Amazon S3 console, AWS Explorer for Visual Studio (Windows), or AWS Toolkit for Visual Studio Code. You can also generate a presigned URL programmatically by using the AWS Command Line Interface (AWS CLI) or the AWS SDKs.

Using the S3 console

You can use the Amazon S3 console to generate a presigned URL for sharing an object up to 5 TB by following these steps. When using the console the maximum expiration time for a presigned URL is 12 hours from the time of creation.

To generate a presigned URL by using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the **General purpose buckets** list, choose the name of the general purpose bucket that contains the object that you want a presigned URL for.
4. In the **Objects** list, select the object that you want to create a presigned URL for.
5. On the **Object actions** menu, choose **Share with a presigned URL**.
6. Specify how long you want the presigned URL to be valid.
7. Choose **Create presigned URL**.
8. When a confirmation appears, the URL is automatically copied to your clipboard. You will see a button to copy the presigned URL if you need to copy it again.

Using the AWS CLI

The following example AWS CLI command generates a presigned URL for sharing an object from an Amazon S3 bucket. When you use the AWS CLI, the maximum expiration time for a presigned URL is 7 days from the time of creation. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3 presign s3://amzn-s3-demo-bucket/mydoc.txt --expires-in 604800
```

The `--expires-in` parameter specifies the expiration time in seconds.

> **Note**
>
> For all AWS Regions launched after March 20, 2019 you need to specify the `endpoint-url` and `AWS Region` with the request. For a list of all the Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.

```
aws s3 presign s3://amzn-s3-demo-bucket/mydoc.txt --expires-in 604800 --region af-south-1 --endpoint-url https://s3.af-south-1.amazonaws.com
```

For more information, see presign in the *AWS CLI Command Reference*.

Using the AWS SDKs

You can generate a presigned URL programmatically by using the AWS SDKs.

Python

The following Python script generates a `GET` presigned URL for sharing an object.

1. Copy the contents of the script and save it as "*get-only-url.py*" file. To use the following examples, replace the *user input placeholders* with your own information (such as your file name).

```
import argparse
import boto3
from botocore.exceptions import ClientError

def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
```

```
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method,
            Params=method_parameters,
            ExpiresIn=expires_in
        )
    except ClientError:
        print(f"Couldn't get a presigned URL for client method
 '{client_method}'.")
        raise
    return url

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key", help="The key (path and filename) in the S3 bucket.",
    )
    args = parser.parse_args()

    # By default, this will use credentials from ~/.aws/credentials
    s3_client = boto3.client("s3")

    # The presigned URL is specified to expire in 1000 seconds
    url = generate_presigned_url(
        s3_client,
        "get_object",
        {"Bucket": args.bucket, "Key": args.key},
        1000
    )
    print(f"Generated GET presigned URL: {url}")

if __name__ == "__main__":
    main()
```

2. To generate a GET presigned URL for sharing a file, run the following script with your bucket name and desired object path.

   The following command uses example values. Replace the *user input placeholders* with your own information.

```
python get-only-url.py amzn-s3-demo-bucket <object-path>
```

The script will output a GET presigned URL:

```
Generated GET presigned URL: https://amzn-s3-demo-
bucket.s3.amazonaws.com/object.txt?
AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=vjbyNxybdZaMmLa
%2ByT372YEAiv4%3D&Expires=1741978496
```

3. You can download the file using the generated presigned URL with curl:

```
curl -X GET "generated-presigned-url" -o "path/to/save/file"
```

For more examples of using the AWS SDKs to generate a presigned URL for sharing an object, see Create a presigned URL for Amazon S3 by using an AWS SDK.

> **Note**
>
> For all AWS Regions launched after March 20, 2019 you need to specify the `endpoint-url` and `AWS Region` with the request. For a list of all the Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.

> **Note**
>
> When using the AWS SDKs, the Tagging attribute must be a header and not a query parameter. All other attributes can be passed as a parameter for the presigned URL.

Using the AWS Toolkit for Visual Studio (Windows)

> **Note**
>
> At this time, the AWS Toolkit for Visual Studio does not support Visual Studio for Mac.

1. Install the AWS Toolkit for Visual Studio using the following instructions, Installing and setting up the Toolkit for Visual Studio in the *AWS Toolkit for Visual Studio User Guide*.

2. Connect to AWS using the following steps, Connecting to AWS in the *AWS Toolkit for Visual Studio User Guide*.
3. In the left side panel labeled **AWS Explorer**, open the bucket containing your object.
4. Open the context menu for the object you wish to have a presigned URL generated for and select **Create Pre-Signed URL...**.
5. In the pop-up window, set the expiration date and time for your presigned URL.
6. The **Object Key**, should pre-populate based on the object you selected.
7. Choose **GET** to specify that this presigned URL will be used for downloading an object.
8. Choose the **Generate** button.
9. To copy the URL to the clipboard, choose **Copy**.
10. To use the generated presigned URL, paste the URL into any browser.

Using AWS Toolkit for Visual Studio Code

If you're using Visual Studio Code, you can generate a presigned URL to share an object without writing any code by using AWS Toolkit for Visual Studio Code. For general information, see AWS Toolkit for Visual Studio Code in the *AWS Toolkit for Visual Studio Code User Guide*.

For instructions on how to install the AWS Toolkit for Visual Studio Code, see Installing the AWS Toolkit for Visual Studio Code in the *AWS Toolkit for Visual Studio Code User Guide*.

1. Connect to AWS using the following steps, Connecting to AWS Toolkit for Visual Studio Code in the *AWS Toolkit for Visual Studio Code User Guide*.
2. Select the AWS logo on the left panel in Visual Studio Code.
3. Under **EXPLORER**, select **S3**.
4. Choose a bucket and file and open the context menu (right-click).
5. Choose **Generate presigned URL**, and then set the expiration time (in minutes).
6. Press Enter, and the presigned URL will be copied to your clipboard.

Uploading objects with presigned URLs

You may use presigned URLs to allow someone to upload an object to your Amazon S3 bucket. Using a presigned URL will allow an upload without requiring another party to have AWS security

credentials or permissions. A presigned URL is limited by the permissions of the user who creates it. That is, if you receive a presigned URL to upload an object, you can upload an object only if the creator of the URL has the necessary permissions to upload that object.

When someone uses the URL to upload an object, Amazon S3 creates the object in the specified bucket. If an object with the same key that is specified in the presigned URL already exists in the bucket, Amazon S3 replaces the existing object with the uploaded object. After upload, the bucket owner will own the object.

For general information about presigned URLs, see Download and upload objects with presigned URLs.

You can create a presigned URL for uploading an object without writing any code by using AWS Explorer for Visual Studio. You can also generate a presigned URL programmatically by using the AWS SDKs.

> **Note**
>
> At this time, the AWS Toolkit for Visual Studio doesn't support Visual Studio for Mac.

Using the AWS Toolkit for Visual Studio (Windows)

1. Install the AWS Toolkit for Visual Studio using the following instructions, Installing and setting up the Toolkit for Visual Studio in the *AWS Toolkit for Visual Studio User Guide*.
2. Connect to AWS using the following steps, Connecting to AWS in the *AWS Toolkit for Visual Studio User Guide*.
3. In the left side panel labeled **AWS Explorer**, open the context menu for the bucket you wish to have an object uploaded to.
4. Choose **Create Pre-Signed URL...**.
5. In the pop-up window, set the expiration date and time for your presigned URL.
6. For **Object Key**, set the name of the file to be uploaded. The file you're uploading must match this name exactly. If an object with the same object key already exists in the bucket, Amazon S3 will replace the existing object with the newly uploaded object.
7. Choose **PUT** to specify that this presigned URL will be used for uploading an object.
8. Choose the **Generate** button.
9. To copy the URL to the clipboard, choose **Copy**.

10. To use this URL you can send a PUT request with the `curl` command. Include the full path to your file and the presigned URL itself.

```
curl -X PUT -T "/path/to/file" "presigned URL"
```

Using the AWS SDKs to generate a PUT presigned URL for uploading a file

You can generate a presigned URL that can perform an S3 action for a limited time.

> **Note**
>
> If you use the AWS CLI or AWS SDKs, the expiration time for presigned URLs can be set as high as 7 days. For more information, see Expiration time for presigned URLs.

Python

The following Python script generates a PUT presigned URL for uploading an object to an S3 general purpose bucket.

1. Copy the contents of the script and save it as "*put-only-url.py*" file. To use the following examples, replace the *user input placeholders* with your own information (such as your file name).

```
import argparse
import boto3
from botocore.exceptions import ClientError

def generate_presigned_url(s3_client, client_method, method_parameters,
 expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
```

```
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method,
            Params=method_parameters,
            ExpiresIn=expires_in
        )
    except ClientError:
        print(f"Couldn't get a presigned URL for client method
 '{client_method}'.")
        raise
    return url

def main():
    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key", help="The key (path and filename) in the S3 bucket.",
    )
    parser.add_argument(
        "--region", help="The AWS region where the bucket is located.",
 default="us-east-1"
    )
    parser.add_argument(
        "--content-type", help="The content type of the file to upload.",
 default="application/octet-stream"
    )
    args = parser.parse_args()

    # Create S3 client with explicit region configuration
    s3_client = boto3.client("s3", region_name=args.region)

    # Optionally set signature version if needed for older S3 regions
    # s3_client.meta.config.signature_version = 's3v4'

    # The presigned URL is specified to expire in 1000 seconds
    url = generate_presigned_url(
        s3_client,
        "put_object",
        {
            "Bucket": args.bucket,
            "Key": args.key,
            "ContentType": args.content_type  # Specify content type
        },
        1000
    )
```

```
    print(f"Generated PUT presigned URL: {url}")

if __name__ == "__main__":
    main()
```

2. To generate a PUT presigned URL for uploading a file, run the following script with your bucket name and desired object path.

   The following command uses example values. Replace the *user input placeholders* with your own information.

   ```
   python put-only-url.py amzn-s3-demo-bucket <object-path> --region us-east-1 --
   content-type application/octet-stream
   ```

   The script will output a PUT presigned URL:

   ```
   Generated PUT presigned URL: https://amzn-s3-demo-
   bucket.s3.amazonaws.com/object.txt?
   AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Signature=vjbyNxybdZaMmLa
   %2ByT372YEAiv4%3D&Expires=1741978496
   ```

3. You can now upload the file using the generated presigned URL with curl. Make sure to include the same content type that was used when generating the URL:

   ```
   curl -X PUT -T "path/to/your/local/file" -H "Content-Type: application/octet-
   stream" "generated-presigned-url"
   ```

   If you specified a different content type when generating the URL, make sure to use that same content type in the curl command.

For more examples of using the AWS SDKs to generate a presigned URL for uploading an object, see Create a presigned URL for Amazon S3 by using an AWS SDK.

Troubleshooting SignatureDoesNotMatch errors

If you encounter a `SignatureDoesNotMatch` error when using presigned URLs, check the following:

- Verify your system time is synchronized with a reliable time server

- Ensure you're using the URL exactly as generated, without any modifications
- Check if the URL has expired and generate a new one if needed
- Make sure the content type in your upload request matches the content type specified when generating the URL
- Confirm you're using the correct region for the bucket
- When using curl, enclose the URL in quotes to properly handle special characters

Transforming objects with S3 Object Lambda

Note

As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

With Amazon S3 Object Lambda, you can add your own code to Amazon S3 `GET`, `LIST`, and `HEAD` requests to modify and process data as it is returned to an application. You can use custom code to modify the data returned by S3 `GET` requests to filter rows, dynamically resize and watermark images, redact confidential data, and more. You can also use S3 Object Lambda to modify the output of S3 `LIST` requests to create a custom view of all objects in a bucket and S3 `HEAD` requests to modify object metadata such as object name and size. You can use S3 Object Lambda as an origin for your Amazon CloudFront distribution to tailor data for end users, such as automatically resizing images, transcoding older formats (like from JPEG to WebP), or stripping metadata. For more information, see the AWS Blog post Use Amazon S3 Object Lambda with Amazon CloudFront. Powered by AWS Lambda functions, your code runs on infrastructure that is fully managed by AWS. Using S3 Object Lambda reduces the need to create and store derivative copies of your data or to run proxies, all with no need to change your applications.

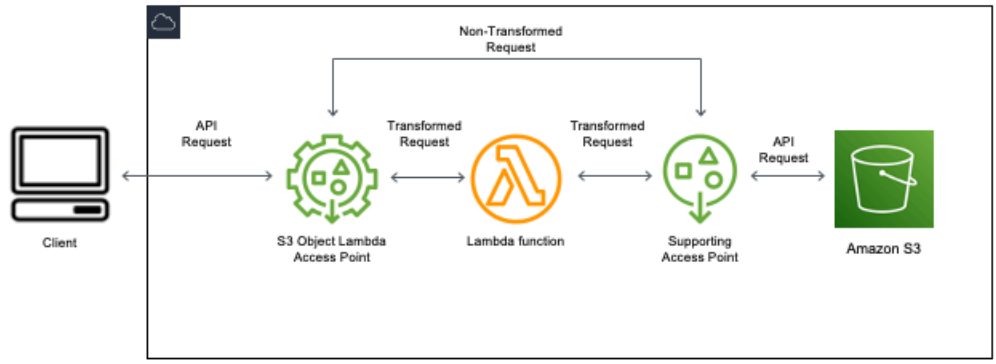
How S3 Object Lambda works

S3 Object Lambda uses AWS Lambda functions to automatically process the output of standard S3 `GET`, `LIST`, or `HEAD` requests. AWS Lambda is a serverless compute service that runs customer-defined code without requiring management of underlying compute resources. You can author

and run your own custom Lambda functions, tailoring the data transformation to your specific use cases.

After you configure a Lambda function, you attach it to an S3 Object Lambda service endpoint, known as an *Object Lambda Access Point*. The Object Lambda Access Point uses a standard S3 access point, known as a *supporting access point*, to access data.

When you send a request to your Object Lambda Access Point, Amazon S3 automatically calls your Lambda function. Any data retrieved by using an S3 `GET`, `LIST`, or `HEAD` request through the Object Lambda Access Point returns a transformed result back to the application. All other requests are processed as normal, as illustrated in the following diagram.



The topics in this section describe how to work with S3 Object Lambda.

Topics

- Creating Object Lambda Access Points
- Using Amazon S3 Object Lambda Access Points
- Security considerations for S3 Object Lambda Access Points
- Writing Lambda functions for S3 Object Lambda Access Points
- Using AWS built Lambda functions
- Best practices and guidelines for S3 Object Lambda
- S3 Object Lambda tutorials

- Debugging and troubleshooting S3 Object Lambda

Creating Object Lambda Access Points

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

An Object Lambda Access Point is associated with exactly one standard access point, which you specify during creation. To create an Object Lambda Access Point, you need the following resources:

- **A standard S3 access point.** When you're working with Object Lambda Access Points, this standard access point is known as a *supporting access point* and is attached to an S3 bucket or Amazon FSx for OpenZFS volume. For information about creating standard access points, see the section called "Creating an access point".
- **An AWS Lambda function.** You can either create your own Lambda function, or you can use a prebuilt function. For more information about creating Lambda functions, see the section called "Writing Lambda functions". For more information about prebuilt functions, see Using AWS built Lambda functions.
- **(Optional) An AWS Identity and Access Management (IAM) policy.** Amazon S3 access points support IAM resource policies that you can use to control the use of the access point by resource, user, or other conditions. For more information about creating these policies, see the section called "Configuring IAM policies".

The following sections describe how to create an Object Lambda Access Point by using:

- The AWS Management Console
- The AWS Command Line Interface (AWS CLI)
- An AWS CloudFormation template
- The AWS Cloud Development Kit (AWS CDK)

For information about how to create an Object Lambda Access Point by using the REST API, see `CreateAccessPointForObjectLambda` in the *Amazon Simple Storage Service API Reference*.

Create an Object Lambda Access Point

Use one of the following procedures to create your Object Lambda Access Point.

Using the S3 console

To create an Object Lambda Access Point by using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to switch to.
3. In the left navigation pane, choose **Object Lambda Access Points**.
4. On the **Object Lambda Access Points** page, choose **Create Object Lambda Access Point**.
5. For **Object Lambda Access Point name**, enter the name that you want to use for the access point.

   As with standard access points, there are rules for naming Object Lambda Access Points. For more information, see Naming rules for access points.
6. For **Supporting Access Point**, enter or browse to the standard access point that you want to use. The access point must be in the same AWS Region as the objects that you want to transform. For information about creating standard access points, see the section called "Creating an access point".
7. Under **Transformation configuration**, you can add a function that transforms your data for your Object Lambda Access Point. Do one of the following:

   - If you already have a AWS Lambda function in your account you can choose it under **Invoke Lambda function**. Here you may enter the Amazon Resource Name (ARN) of an Lambda function in your AWS account or choose a Lambda function from the drop-down menu.
   - If you wish to use a AWS built function choose the function name under **AWS built function** and select **Create Lambda function**. This will take you to the Lambda console where you can deploy a built function into your AWS account. For more information about built functions, see Using AWS built Lambda functions.

Under **S3 APIs**, choose one or more API operations to invoke. For each API selected you must specify a Lambda function to invoke.

8. (Optional) Under **Payload**, add JSON text that you want to provide to your Lambda function as input. You can configure payloads with different parameters for different Object Lambda Access Points that invoke the same Lambda function, thereby extending the flexibility of your Lambda function.

   > **Important**
   >
   > When you're using Object Lambda Access Points, make sure that the payload does not contain any confidential information.

9. (Optional) For **Range and part number**, you must enable this option if you want to process `GET` and `HEAD` requests with range and part number headers. Enabling this option confirms that your Lambda function can recognize and process these requests. For more information about range headers and part numbers, see Working with Range and partNumber headers.
10. (Optional) For **Request metrics**, choose **Enable** or **Disable** to add Amazon S3 monitoring to your Object Lambda Access Point. Request metrics are billed at the standard Amazon CloudWatch rate.
11. (Optional) Under **Object Lambda Access Point policy**, set a resource policy. Resource policies grant permissions for the specified Object Lambda Access Point and can control the use of the access point by resource, user, or other conditions. For more information about Object Lambda Access Point resource policies see, Configuring IAM policies for Object Lambda Access Points.
12. Under **Block Public Access settings for this Object Lambda Access Point**, select the block public access settings that you want to apply. All block public access settings are enabled by default for new Object Lambda Access Points, and we recommend that you leave default settings enabled. Amazon S3 currently doesn't support changing an Object Lambda Access Point's block public access settings after the Object Lambda Access Points has been created.

    For more information about using Amazon S3 Block Public Access, see Managing public access to access points for general purpose buckets.
13. Choose **Create Object Lambda Access Point**.

Using the AWS CLI

To create an Object Lambda Access Point by using an AWS CloudFormation template

> **Note**
>
> To use the following commands, replace the *user input placeholders* with your own information.

1. Download the AWS Lambda function deployment package `s3objectlambda_deployment_package.zip` at S3 Object Lambda default configuration.
2. Run the following `put-object` command to upload the package to an Amazon S3 bucket.

```
aws s3api put-object --bucket Amazon S3 bucket name --key
 s3objectlambda_deployment_package.zip --body release/
s3objectlambda_deployment_package.zip
```

3. Download the AWS CloudFormation template `s3objectlambda_defaultconfig.yaml` at S3 Object Lambda default configuration.
4. Run the following `deploy` command to deploy the template to your AWS account.

```
aws cloudformation deploy --template-file s3objectlambda_defaultconfig.yaml \
 --stack-name CloudFormation stack name \
 --parameter-overrides ObjectLambdaAccessPointName=Object Lambda Access Point name
 \
  SupportingAccessPointName=Amazon S3 access point S3BucketName=Amazon S3 bucket \
  LambdaFunctionS3BucketName=Amazon S3 bucket containing your Lambda package \
  LambdaFunctionS3Key=Lambda object key LambdaFunctionS3ObjectVersion=Lambda object
 version \
  LambdaFunctionRuntime=Lambda function runtime --capabilities capability_IAM
```

You can configure this AWS CloudFormation template to invoke Lambda for `GET`, `HEAD`, and `LIST` API operations. For more information about modifying the template's default configuration, see the section called “Automate S3 Object Lambda setup with AWS CloudFormation”.

To create an Object Lambda Access Point by using the AWS CLI

> **Note**
>
> To use the following commands, replace the *`user input placeholders`* with your own information.

The following example creates an Object Lambda Access Point named *`my-object-lambda-ap`* for the bucket *`amzn-s3-demo-bucket1`* in the account *`111122223333`*. This example assumes that a standard access point named *`example-ap`* has already been created. For information about creating a standard access point, see the section called "Creating an access point".

This example uses the AWS prebuilt function `decompress`. For more information about prebuilt functions, see the section called "Using AWS built functions".

1. Create a bucket. In this example, we will use *`amzn-s3-demo-bucket1`*. For information about creating buckets, see the section called "Creating a general purpose bucket".
2. Create a standard access point and attach it to your bucket. In this example, we will use *`example-ap`*. For information about creating standard access points, see the section called "Creating an access point".
3. Do one of the following:

   - Create a Lambda function in your account that you would like to use to transform your Amazon S3 object. For more information about creating Lambda functions, see the section called "Writing Lambda functions". To use your custom function with the AWS CLI, see Using Lambda with the AWS CLI in the *AWS Lambda Developer Guide*.
   - Use an AWS prebuilt Lambda function. For more information about prebuilt functions, see Using AWS built Lambda functions.
4. Create a JSON configuration file named `my-olap-configuration.json`. In this configuration, provide the supporting access point and the Amazon Resource Name (ARN) for the Lambda function that you created in the previous steps or the ARN for the prebuilt function that you're using.

   Example

```
{
```

```
    "SupportingAccessPoint" : "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
    "TransformationConfigurations": [{
        "Actions" : ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation" : {
            "AwsLambda": {
                "FunctionPayload" : "{\"compressionType\":\"gzip\"}",
                "FunctionArn" : "arn:aws:lambda:us-east-1:111122223333:function/
compress"
            }
        }
    }]
}
```

5. Run the `create-access-point-for-object-lambda` command to create your Object Lambda Access Point.

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-ap --configuration file://my-olap-configuration.json
```

6. (Optional) Create a JSON policy file named `my-olap-policy.json`.

   Adding an Object Lambda Access Point resource policy can control the use of the access point by resource, user, or other conditions. This resource policy grants the `GetObject` permission for account *444455556666* to the specified Object Lambda Access Point.

   Example

```
{
    "Version": "2008-10-17",
    "Statement": [
        {
            "Sid": "Grant account 444455556666 GetObject access",
            "Effect": "Allow",
            "Action": "s3-object-lambda:GetObject",
            "Principal": {
                "AWS": "arn:aws:iam::444455556666:root"
            },
            "Resource": "your-object-lambda-access-point-arn"
        }
    ]
```

```
}
```

7. (Optional) Run the `put-access-point-policy-for-object-lambda` command to set your resource policy.

```
aws s3control put-access-point-policy-for-object-lambda --account-id 111122223333
 --name my-object-lambda-ap --policy file://my-olap-policy.json
```

8. (Optional) Specify a payload.

   A payload is optional JSON that you can provide to your AWS Lambda function as input. You can configure payloads with different parameters for different Object Lambda Access Points that invoke the same Lambda function, thereby extending the flexibility of your Lambda function.

   The following Object Lambda Access Point configuration shows a payload with two parameters.

```
{
 "SupportingAccessPoint": "AccessPointArn",
 "CloudWatchMetricsEnabled": false,
 "TransformationConfigurations": [{
  "Actions": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
  "ContentTransformation": {
   "AwsLambda": {
    "FunctionArn": "FunctionArn",
    "FunctionPayload": "{\"res-x\": \"100\",\"res-y\": \"100\"}"
   }
  }
 }]
}
```

   The following Object Lambda Access Point configuration shows a payload with one parameter, and with `GetObject-Range`, `GetObject-PartNumber`, `HeadObject-Range`, and `HeadObject-PartNumber` enabled.

```
{
    "SupportingAccessPoint":"AccessPointArn",
    "CloudWatchMetricsEnabled": false,
    "AllowedFeatures": ["GetObject-Range", "GetObject-PartNumber", "HeadObject-
Range", "HeadObject-PartNumber"],
```

```
    "TransformationConfigurations": [{
        "Action": ["GetObject", "HeadObject", "ListObjects", "ListObjectsV2"],
        "ContentTransformation": {
            "AwsLambda": {
                "FunctionArn":"FunctionArn",
                "FunctionPayload": "{\"compression-amount\": \"5\"}"
            }
        }
    }]
}
```

> **Important**
>
> When you're using Object Lambda Access Points, make sure that the payload does not contain any confidential information.

Using the AWS CloudFormation console and template

You can create an Object Lambda Access Point by using the default configuration provided by Amazon S3. You can download an AWS CloudFormation template and Lambda function source code from the GitHub repository and deploy these resources to set up a functional Object Lambda Access Point.

For information about modifying the AWS CloudFormation template's default configuration, see the section called “Automate S3 Object Lambda setup with AWS CloudFormation”.

For information about configuring Object Lambda Access Points by using CloudFormation without the template, see `AWS::S3ObjectLambda::AccessPoint` in the *AWS CloudFormation User Guide*.

To upload the Lambda function deployment package

1. Download the AWS Lambda function deployment package `s3objectlambda_deployment_package.zip` at S3 Object Lambda default configuration.
2. Upload the package to an Amazon S3 bucket.

To create an Object Lambda Access Point by using the AWS CloudFormation console

1. Download the AWS CloudFormation template `s3objectlambda_defaultconfig.yaml` at S3 Object Lambda default configuration.
2. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
3. Do one of the following:
   - If you've never used AWS CloudFormation before, on the AWS CloudFormation home page, choose **Create stack**.
   - If you have used AWS CloudFormation before, in the left navigation pane, choose **Stacks**. Choose **Create stack**, then choose **With new resources (standard)**.
4. For **Prerequisite - Prepare template**, choose **Template is ready**.
5. For **Specify template**, choose **Upload a template file** and upload `s3objectlambda_defaultconfig.yaml`.
6. Choose **Next**.
7. On the **Specify stack details** page, enter a name for the stack.
8. In the **Parameters** section, specify the following parameters that are defined in the stack template:
   a. For **CreateNewSupportingAccessPoint**, do one of the following:
      - If you already have a supporting access point for the S3 bucket where you uploaded the template, choose **false**.
      - If you want to create a new access point for this bucket, choose **true**.
   b. For **EnableCloudWatchMonitoring**, choose **true** or **false**, depending on whether you want to enable Amazon CloudWatch request metrics and alarms.
   c. (Optional) For **LambdaFunctionPayload**, add JSON text that you want to provide to your Lambda function as input. You can configure payloads with different parameters for different Object Lambda Access Points that invoke the same Lambda function, thereby extending the flexibility of your Lambda function.

> **Important**
>
> When you're using Object Lambda Access Points, make sure that the payload does not contain any confidential information.

d. For **LambdaFunctionRuntime**, enter your preferred runtime for the Lambda function. The available choices are `nodejs14.x`, `python3.9`, `java11`.

e. For **LambdaFunctionS3BucketName**, enter the Amazon S3 bucket name where you uploaded the deployment package.

f. For **LambdaFunctionS3Key**, enter the Amazon S3 object key where you uploaded the deployment package.

g. For **LambdaFunctionS3ObjectVersion**, enter the Amazon S3 object version where you uploaded the deployment package.

h. For **ObjectLambdaAccessPointName**, enter a name for your Object Lambda Access Point.

i. For **S3BucketName**, enter the Amazon S3 bucket name that will be associated with your Object Lambda Access Point.

j. For **SupportingAccessPointName**, enter the name of your supporting access point.

> **Note**
>
> This is an access point that is associated with the Amazon S3 bucket that you chose in the previous step. If you do not have any access points associated with your Amazon S3 bucket, you can configure the template to create one for you by choosing **true** for **CreateNewSupportingAccessPoint**.

9. Choose **Next**.
10. On the **Configure stack options page**, choose **Next**.

    For more information about the optional settings on this page, see Setting AWS CloudFormation stack options in the *AWS CloudFormation User Guide*.
11. On the **Review** page, choose **Create stack**.

Using the AWS Cloud Development Kit (AWS CDK)

For more information about configuring Object Lambda Access Points by using the AWS CDK, see `AWS::S3ObjectLambda` Construct Library in the *AWS Cloud Development Kit (AWS CDK) API Reference*.

Automate S3 Object Lambda setup with a CloudFormation template

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

You can use an AWS CloudFormation template to quickly create an Amazon S3 Object Lambda Access Point. The CloudFormation template automatically creates relevant resources, configures AWS Identity and Access Management (IAM) roles, and sets up an AWS Lambda function that automatically handles requests through the Object Lambda Access Point. With the CloudFormation template, you can implement best practices, improve your security posture, and reduce errors caused by manual processes.

This GitHub repository contains the CloudFormation template and Lambda function source code. For instructions on how to use the template, see the section called “Creating Object Lambda Access Points”.

The Lambda function provided in the template does not run any transformation. Instead, it returns your objects as-is from the underlying data source. You can clone the function and add your own transformation code to modify and process data as it is returned to an application. For more information about modifying your function, see the section called “Modifying the Lambda function” and the section called “Writing Lambda functions”.

Modifying the template

Creating a new supporting access point

S3 Object Lambda uses two access points, an Object Lambda Access Point and a standard S3 access point, which is referred to as the *supporting access point*. When you make a request to an Object Lambda Access Point, S3 either invokes Lambda on your behalf, or it delegates the

request to the supporting access point, depending upon the S3 Object Lambda configuration. You can create a new supporting access point by passing the following parameter as part of the `aws cloudformation deploy` command when deploying the template.

```
CreateNewSupportingAccessPoint=true
```

Configuring a function payload

You can configure a payload to provide supplemental data to the Lambda function by passing the following parameter as part of the `aws cloudformation deploy` command when deploying the template.

```
LambdaFunctionPayload="format=json"
```

Enabling Amazon CloudWatch monitoring

You can enable CloudWatch monitoring by passing the following parameter as part of the `aws cloudformation deploy` command when deploying the template.

```
EnableCloudWatchMonitoring=true
```

This parameter enables your Object Lambda Access Point for Amazon S3 request metrics and creates two CloudWatch alarms to monitor client-side and server-side errors.

> **Note**
>
> Amazon CloudWatch usage will incur additional costs. For more information about Amazon S3 request metrics, see Monitoring and logging access points.
> For pricing details, see CloudWatch pricing.

Configuring provisioned concurrency

To reduce latency, you can configure provisioned concurrency for the Lambda function that's backing the Object Lambda Access Point by editing the template to include the following lines under `Resources`.

```
LambdaFunctionVersion:
      Type: AWS::Lambda::Version
```

```
      Properties:
        FunctionName: !Ref LambdaFunction
        ProvisionedConcurrencyConfig:
            ProvisionedConcurrentExecutions: Integer
```

Note

You will incur additional charges for provisioning concurrency. For more information about provisioned concurrency, see Managing Lambda provisioned concurrency in the *AWS Lambda Developer Guide*.
For pricing details, see AWS Lambda pricing.

Modifying the Lambda function

Changing header values for a `GetObject` request

By default, the Lambda function forwards all headers, except `Content-Length` and `ETag`, from the presigned URL request to the `GetObject` client. Based on your transformation code in the Lambda function, you can choose to send new header values to the `GetObject` client.

You can update your Lambda function to send new header values by passing them in the `WriteGetObjectResponse` API operation.

For example, if your Lambda function translates text in Amazon S3 objects to a different language, you can pass a new value in the `Content-Language` header. You can do this by modifying the `writeResponse` function as follows:

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
    },
```

```
    ...headers,
    ContentLanguage: 'my-new-language'
  }).promise();
}
```

For a full list of supported headers, see `WriteGetObjectResponse` in the *Amazon Simple Storage Service API Reference*.

Returning metadata headers

You can update your Lambda function to send new header values by passing them in the `WriteGetObjectResponse` API operation request.

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest,
      'my-new-header': 'my-new-value'
    },
    ...headers
  }).promise();
}
```

Returning a new status code

You can return a custom status code to the `GetObject` client by passing it in the `WriteGetObjectResponse` API operation request.

```
async function writeResponse (s3Client: S3, requestContext: GetObjectContext,
 transformedObject: Buffer,
  headers: Headers): Promise<PromiseResult<{}, AWSError>> {
  const { algorithm, digest } = getChecksum(transformedObject);

  return s3Client.writeGetObjectResponse({
```

```
    RequestRoute: requestContext.outputRoute,
    RequestToken: requestContext.outputToken,
    Body: transformedObject,
    Metadata: {
      'body-checksum-algorithm': algorithm,
      'body-checksum-digest': digest
    },
    ...headers,
    StatusCode: Integer
  }).promise();
}
```

For a full list of supported status codes, see `WriteGetObjectResponse` in the *Amazon Simple Storage Service API Reference*.

Applying Range and `partNumber` parameters to the source object

By default, the Object Lambda Access Point created by the CloudFormation template can handle the `Range` and `partNumber` parameters. The Lambda function applies the range or part number requested to the transformed object. To do so, the function must download the whole object and run the transformation. In some cases, your transformed object ranges might map exactly to your source object ranges. This means that requesting byte range A-B on your source object and running the transformation might produce the same result as requesting the whole object, running the transformation, and returning byte range A-B on the transformed object.

In such cases, you can change the Lambda function implementation to apply the range or part number directly to the source object. This approach reduces the overall function latency and memory required. For more information, see the section called “Working with Range and partNumber headers”.

Disabling Range and `partNumber` handling

By default, the Object Lambda Access Point created by the CloudFormation template can handle the `Range` and `partNumber` parameters. If you don't need this behavior, you can disable it by removing the following lines from the template:

```
AllowedFeatures:
  - GetObject-Range
  - GetObject-PartNumber
  - HeadObject-Range
  - HeadObject-PartNumber
```

Transforming large objects

By default, the Lambda function processes the entire object in memory before it can start streaming the response to S3 Object Lambda. You can modify the function to stream the response as it performs the transformation. Doing so helps reduce the transformation latency and the Lambda function memory size. For an example implementation, see the Stream compressed content example.

Using Amazon S3 Object Lambda Access Points

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

Making requests through Amazon S3 Object Lambda Access Points works the same as making requests through other access points. For more information about how to make requests through an access point, see Using Amazon S3 access points for general purpose buckets. You can make requests through Object Lambda Access Points by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

> **Important**
>
> The Amazon Resource Names (ARNs) for Object Lambda Access Points use a service name of `s3-object-lambda`. Thus, Object Lambda Access Point ARNs begin with `arn:aws::s3-object-lambda`, instead of `arn:aws::s3`, which is used with other access points.

How to find the ARN for your Object Lambda Access Point

To use an Object Lambda Access Point with the AWS CLI or AWS SDKs, you need to know the Amazon Resource Name (ARN) of the Object Lambda Access Point. The following examples show how to find the ARN for an Object Lambda Access Point by using the Amazon S3 console or AWS CLI.

Using the S3 console

To find the ARN for your Object Lambda Access Point by using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, under **Access management and security**, choose **Access Points**.
3. Choose the **Object Lambda** tab.
4. Choose the option button next to the Object Lambda Access Point whose ARN you want to copy.
5. Choose **Copy ARN**.

Using the AWS CLI

To find the ARN for your Object Lambda Access Point by using the AWS CLI

1. To retrieve a list of the Object Lambda Access Points that are associated with your AWS account, run the following command. Before running the command, replace the account ID *111122223333* with your AWS account ID.

```
aws s3control list-access-points-for-object-lambda --account-id 111122223333
```

2. Review the command output to find the Object Lambda Access Point ARN that you want to use. The output of the previous command should look similar to the following example.

```
{
    "ObjectLambdaAccessPointList": [
        {
            "Name": "my-object-lambda-ap",
            "ObjectLambdaAccessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/my-object-lambda-ap"
        },
        ...
    ]
}
```

How to use a bucket-style alias for your S3 bucket Object Lambda Access Point

When you create an Object Lambda Access Point, Amazon S3 automatically generates a unique alias for your Object Lambda Access Point. You can use this alias instead of an Amazon S3 bucket name or the Object Lambda Access Point Amazon Resource Name (ARN) in a request for access point data plane operations. For a list of these operations, see Access point compatibility.

An Object Lambda Access Point alias name is created within the same namespace as an Amazon S3 bucket. This alias name is automatically generated and cannot be changed. For an existing Object Lambda Access Point, an alias is automatically assigned for use. An Object Lambda Access Point alias name meets all the requirements of a valid Amazon S3 bucket name and consists of the following parts:

Object Lambda Access Point name prefix-*metadata*--ol-s3

> **Note**
>
> The --ol-s3 suffix is reserved for Object Lambda Access Point alias names and can't be used for bucket or Object Lambda Access Point names. For more information about Amazon S3 bucket-naming rules, see General purpose bucket naming rules.

The following examples show the ARN and the Object Lambda Access Point alias for an Object Lambda Access Point named *my-object-lambda-access-point*:

- **ARN** – arn:aws:s3-object-lambda:*region*:*account-id*:accesspoint/*my-object-lambda-access-point*
- **Object Lambda Access Point alias** – *my-object-lambda-acc*-*1a4n8yjrb3kda96f67zwrwiiuse1a*--ol-s3

When you use an Object Lambda Access Point, you can use the Object Lambda Access Point alias name without requiring extensive code changes.

When you delete an Object Lambda Access Point, the Object Lambda Access Point alias name becomes inactive and unprovisioned.

How to find the alias for your Object Lambda Access Point

Using the S3 console

To find the alias for your Object Lambda Access Point by using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, under **Access management and security**, choose **Access Points**.
3. Choose the **Object Lambda** tab.
4. For the Object Lambda Access Point that you want to use, copy the **Object Lambda Access Point alias** value.

Using the AWS CLI

When you create an Object Lambda Access Point, Amazon S3 automatically generates an Object Lambda Access Point alias name, as shown in the following example command. To run this command, replace the *`user input placeholders`* with your own information. For information about how to create an Object Lambda Access Point by using the AWS CLI, see To create an Object Lambda Access Point by using the AWS CLI.

```
aws s3control create-access-point-for-object-lambda --account-id 111122223333 --
name my-object-lambda-access-point --configuration file://my-olap-configuration.json
{
    "ObjectLambdaAccessPointArn": "arn:aws:s3:region:111122223333:accesspoint/my-
access-point",
    "Alias": {
        "Value": "my-object-lambda-acc-1a4n8yjrb3kda96f67zwrwiiuse1a--ol-s3",
        "Status": "READY"
    }
}
```

The generated Object Lambda Access Point alias name has two fields:

- The `Value` field is the alias value of the Object Lambda Access Point.
- The `Status` field is the status of the Object Lambda Access Point alias. If the status is `PROVISIONING`, Amazon S3 is provisioning the Object Lambda Access Point alias, and the alias is not yet ready for use. If the status is `READY`, the Object Lambda Access Point alias has been successfully provisioned and is ready for use.

For more information about the `ObjectLambdaAccessPointAlias` data type in the REST API, see CreateAccessPointForObjectLambda and ObjectLambdaAccessPointAlias in the *Amazon Simple Storage Service API Reference*.

How to use the Object Lambda Access Point alias

You can use an Object Lambda Access Point alias instead of an Amazon S3 bucket name for the operations listed in Access point compatibility.

The following AWS CLI example for the `get-bucket-location` command uses the bucket's access point alias to return the AWS Region that the bucket is in. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api get-bucket-location --bucket my-object-lambda-
acc-w7i37nq6xuzgax3jw3oqtifiusw2a--ol-s3

{
    "LocationConstraint": "us-west-2"
}
```

If the Object Lambda Access Point alias in a request isn't valid, the error code `InvalidAccessPointAliasError` is returned. For more information about `InvalidAccessPointAliasError`, see List of Error Codes in the *Amazon Simple Storage Service API Reference*.

The limitations of an Object Lambda Access Point alias are the same as those of an access point alias. For more information about the limitations of an access point alias, see Access point alias limitations.

Security considerations for S3 Object Lambda Access Points

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

With Amazon S3 Object Lambda, you can perform custom transformations on data as it leaves Amazon S3 by using the scale and flexibility of AWS Lambda as a compute platform. S3 and Lambda remain secure by default, but to maintain this security, special consideration by the Lambda function author is required. S3 Object Lambda requires that all access be made by authenticated principals (no anonymous access) and over HTTPS.

To mitigate security risks, we recommend the following:

- Scope the Lambda execution role to the smallest set of permissions possible.
- Whenever possible, make sure your Lambda function accesses Amazon S3 through the provided presigned URL.

Configuring IAM policies

S3 access points support AWS Identity and Access Management (IAM) resource policies that allow you to control the use of the access point by resource, user, or other conditions. For more information, see Configuring IAM policies for Object Lambda Access Points.

Encryption behavior

Because Object Lambda Access Points use both Amazon S3 and AWS Lambda, there are differences in encryption behavior. For more information about default S3 encryption behavior, see Setting default server-side encryption behavior for Amazon S3 buckets.

- When you're using S3 server-side encryption with Object Lambda Access Points, the object is decrypted before being sent to Lambda. After the object is sent to Lambda, it is processed unencrypted (in the case of a `GET` or `HEAD` request).
- To prevent the encryption key from being logged, S3 rejects `GET` and `HEAD` requests for objects that are encrypted by using server-side encryption with customer-provided keys (SSE-C). However, the Lambda function might still retrieve these objects if it has access to the client-provided key.
- When using S3 client-side encryption with Object Lambda Access Points, make sure that Lambda has access to the encryption key so that it can decrypt and re-encrypt the object.

Access points security

S3 Object Lambda uses two access points, an Object Lambda Access Point and a standard S3 access point, which is referred to as the *supporting access point*. When you make a request to an Object

Lambda Access Point, S3 either invokes Lambda on your behalf, or it delegates the request to the supporting access point, depending upon the S3 Object Lambda configuration. When Lambda is invoked for a request S3 generates a presigned URL to your object on your behalf through the supporting access point. Your Lambda function receives this URL as input when the function is invoked.

You can set your Lambda function to use this presigned URL to retrieve the original object, instead of invoking S3 directly. By using this model, you can apply better security boundaries to your objects. You can limit direct object access through S3 buckets or S3 access points to a limited set of IAM roles or users. This approach also protects your Lambda functions from being subject to the confused deputy problem, where a misconfigured function with different permissions than the invoker could allow or deny access to objects when it should not.

Object Lambda Access Point public access

S3 Object Lambda does not allow anonymous or public access because Amazon S3 must authorize your identity to complete any S3 Object Lambda request. When invoking requests through an Object Lambda Access Point, you must have the `lambda:InvokeFunction` permission for the configured Lambda function. Similarly, when invoking other API operations through an Object Lambda Access Point, you must have the required `s3:*` permissions.

Without these permissions, requests to invoke Lambda or delegate to S3 will fail with HTTP 403 (Forbidden) errors. All access must be made by authenticated principals. If you require public access, you can use Lambda@Edge as a possible alternative. For more information, see Customizing at the edge with Lambda@Edge in the *Amazon CloudFront Developer Guide*.

Object Lambda Access Point IP addresses

The `describe-managed-prefix-lists` subnets support gateway virtual private cloud (VPC) endpoints and are related to the routing table of VPC endpoints. Since Object Lambda Access Point does not support gateway VPC its IP ranges are missing. The missing ranges belong to Amazon S3, but are not supported by gateway VPC endpoints. For more information about `describe-managed-prefix-lists`, see DescribeManagedPrefixLists in the *Amazon EC2 API Reference* and AWS IP address ranges in the *AWS General Reference*.

Configuring IAM policies for Object Lambda Access Points

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

Amazon S3 access points support AWS Identity and Access Management (IAM) resource policies that you can use to control the use of the access point by resource, user, or other conditions. You can control access through an optional resource policy on your Object Lambda Access Point, or a resource policy on supporting access point. For step-by-step examples, see Tutorial: Transforming data for your application with S3 Object Lambda and Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend.

The following four resources must have permissions granted to work with Object Lambda Access Points:

- The IAM identity, such as user or role. For more information about IAM identities and best practices, see IAM identities (users, user groups, and roles) in the *IAM User Guide*.
- The standard access point connected to an underlying data source such as an S3 bucket or Amazon FSx for OpenZFS volume. When you're working with Object Lambda Access Points, this standard access point is known as a *supporting access point*.
- The Object Lambda Access Point.
- The AWS Lambda function.

> **Important**
>
> Before you save your policy, make sure to resolve security warnings, errors, general warnings, and suggestions from AWS Identity and Access Management Access Analyzer. IAM Access Analyzer runs policy checks to validate your policy against IAM policy grammar and best practices. These checks generate findings and provide actionable recommendations to help you author policies that are functional and conform to security best practices.

To learn more about validating policies by using IAM Access Analyzer, see IAM Access Analyzer policy validation in the *IAM User Guide*. To view a list of the warnings, errors, and suggestions that are returned by IAM Access Analyzer, see IAM Access Analyzer policy check reference.

The following policy examples assume that you have the following resources:

- An Amazon S3 bucket with the following Amazon Resource Name (ARN):

  `arn:aws:s3:::amzn-s3-demo-bucket1`
- An Amazon S3 standard access point on this bucket with the following ARN:

  `arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point`
- An Object Lambda Access Point with the following ARN:

  `arn:aws:s3-object-lambda:us-east-1:111122223333:accesspoint/my-object-lambda-ap`
- An AWS Lambda function with the following ARN:

  `arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction`

Note

If you're using a Lambda function from your account, you must include the specific function version in your policy statement. In the following example ARN, the version is indicated by *1*:
`arn:aws:lambda:us-east-1:111122223333:function:MyObjectLambdaFunction:1`
Lambda doesn't support adding IAM policies to the version $LATEST. For more information about Lambda function versions, see Lambda function versions in the *AWS Lambda Developer Guide*.

Example– Bucket policy that delegates access control to standard access points

The following S3 bucket policy example delegates access control for a bucket to the bucket's standard access points. This policy allows full access to all access points that are owned by the

bucket owner's account. Thus, all access to this bucket is controlled by the policies that are attached to its access points. Users can read from the bucket only through an access point, which means that operations can be invoked only through access points. For more information, see Delegating access control to access points.

JSON

```
{
    "Version":"2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS":"account-ARN"},
        "Action" : "*",
        "Resource" : [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Bucket owner's
 account ID" }
        }
    }]
}
```

Example– IAM policy that grants a user the necessary permissions to use an Object Lambda Access Point

The following IAM policy grants a user permissions to the Lambda function, the standard access point, and the Object Lambda Access Point.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowLambdaInvocation",
      "Action": [
        "lambda:InvokeFunction"
```

```
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:lambda:us-
east-1:111122223333:function:MyObjectLambdaFunction:1",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowStandardAccessPointAccess",
      "Action": [
        "s3:Get*",
        "s3:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point/
*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": [
            "s3-object-lambda.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "AllowObjectLambdaAccess",
      "Action": [
        "s3-object-lambda:Get*",
        "s3-object-lambda:List*"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/my-object-lambda-ap"
    }
  ]
}
```

Enable permissions for Lambda execution roles

When GET requests are made to an Object Lambda Access Point, your Lambda function needs permission to send data to S3 Object Lambda Access Point. This permission is provided by enabling the `s3-object-lambda:WriteGetObjectResponse` permission on your Lambda function's execution role. You can create a new execution role or update an existing one.

> **Note**
>
> Your function needs the `s3-object-lambda:WriteGetObjectResponse` permission only if you're making a GET request.

To create an execution role in the IAM console

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**.
3. Choose **Create role**.
4. Under **Common use cases**, choose **Lambda**.
5. Choose **Next**.
6. On the **Add permissions** page, search for the AWS managed policy `AmazonS3ObjectLambdaExecutionRolePolicy`, and then select the check box beside the policy name.

   This policy should contain the `s3-object-lambda:WriteGetObjectResponse` Action.
7. Choose **Next**.
8. On the **Name, review, and create** page, for **Role name**, enter **`s3-object-lambda-role`**.
9. (Optional) Add a description and tags for this role.
10. Choose **Create role**.
11. Apply the newly created **`s3-object-lambda-role`** as your Lambda function's execution role. This can be done during or after Lambda function creation in the Lambda console.

For more information about execution roles, see Lambda execution role in the *AWS Lambda Developer Guide*.

Using context keys with Object Lambda Access Points

S3 Object Lambda will evaluate context keys such as `s3-object-lambda:TlsVersion` or `s3-object-lambda:AuthType` that are related to the connection or signing of the request. All other context keys, such as `s3:prefix`, are evaluated by Amazon S3.

Object Lambda Access Point CORS support

When S3 Object Lambda receives a request from a browser or the request includes an `Origin` header, S3 Object Lambda always adds an `"AllowedOrigins":"*"` header field.

For more information, see Using cross-origin resource sharing (CORS).

Writing Lambda functions for S3 Object Lambda Access Points

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

This section details how to write AWS Lambda functions for use with Amazon S3 Object Lambda Access Points.

To learn about complete end-to-end procedures for some S3 Object Lambda tasks, see the following:

- Tutorial: Transforming data for your application with S3 Object Lambda
- Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved

Topics

- Working with GetObject requests in Lambda
- Working with HeadObject requests in Lambda
- Working with ListObjects requests in Lambda
- Working with ListObjectsV2 requests in Lambda

- Event context format and usage
- Working with Range and partNumber headers

Working with `GetObject` requests in Lambda

This section assumes that your Object Lambda Access Point is configured to call the Lambda function for `GetObject`. S3 Object Lambda includes the Amazon S3 API operation, `WriteGetObjectResponse`, which enables the Lambda function to provide customized data and response headers to the `GetObject` caller.

`WriteGetObjectResponse` gives you extensive control over the status code, response headers, and response body, based on your processing needs. You can use `WriteGetObjectResponse` to respond with the whole transformed object, portions of the transformed object, or other responses based on the context of your application. The following section shows unique examples of using the `WriteGetObjectResponse` API operation.

- **Example 1:** Respond with HTTP status code 403 (Forbidden)
- **Example 2:** Respond with a transformed image
- **Example 3:** Stream compressed content

Example 1: Respond with HTTP status code 403 (Forbidden)

You can use `WriteGetObjectResponse` to respond with the HTTP status code 403 (Forbidden) based on the content of the object.

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.WriteGetObjectResponseRequest;

import java.io.ByteArrayInputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
```

```
import java.net.http.HttpResponse;

public class Example1 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        S3Client s3Client = S3Client.builder().build();

        // Check to see if the request contains all of the necessary information.
        // If it does not, send a 4XX response and a custom error code and message.
        // Otherwise, retrieve the object from S3 and stream it
        // to the client unchanged.
        var tokenIsNotPresent = !
event.getUserRequest().getHeaders().containsKey("requiredToken");
        if (tokenIsNotPresent) {
            s3Client.writeGetObjectResponse(WriteGetObjectResponseRequest.builder()
                    .requestRoute(event.outputRoute())
                    .requestToken(event.outputToken())
                    .statusCode(403)
                    .contentLength(0L)
                    .errorCode("MissingRequiredToken")
                    .errorMessage("The required token was not present in the
 request.")
                    .build(),
                    RequestBody.fromInputStream(new ByteArrayInputStream(new
 byte[0]), 0L));
            return;
        }

        // Prepare the presigned URL for use and make the request to S3.
        HttpClient httpClient = HttpClient.newBuilder().build();
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // Stream the original bytes back to the caller.
        s3Client.writeGetObjectResponse(WriteGetObjectResponseRequest.builder()
                .requestRoute(event.outputRoute())
                .requestToken(event.outputToken())
                .build(),
                RequestBody.fromInputStream(presignedResponse.body(),
                    presignedResponse.headers().firstValueAsLong("content-
length").orElse(-1L)));
    }
```

```
}
```

Python

```
import boto3
import requests

def handler(event, context):
    s3 = boto3.client('s3')

    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and contains a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to
    S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    user_request_headers = event["userRequest"]["headers"]

    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    # Check for the presence of a 'CustomHeader' header and deny or allow based on
 that header.
    is_token_present = "SuperSecretToken" in user_request_headers

    if is_token_present:
        # If the user presented our custom 'SuperSecretToken' header, we send the
 requested object back to the user.
        response = requests.get(s3_url)
        s3.write_get_object_response(RequestRoute=route, RequestToken=token,
 Body=response.content)
    else:
        # If the token is not present, we send an error back to the user.
        s3.write_get_object_response(RequestRoute=route, RequestToken=token,
 StatusCode=403,
        ErrorCode="NoSuperSecretTokenFound", ErrorMessage="The request was not
 secret enough.")
```

```
    # Gracefully exit the Lambda function.
    return { 'status_code': 200 }
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and contains a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    // The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    const { userRequest, getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // Check for the presence of a 'CustomHeader' header and deny or allow based on
 that header.
    const isTokenPresent = Object
        .keys(userRequest.headers)
        .includes("SuperSecretToken");

    if (!isTokenPresent) {
        // If the token is not present, we send an error back to the user. The
 'await' in front of the request
        // indicates that we want to wait for this request to finish sending before
 moving on.
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            StatusCode: 403,
            ErrorCode: "NoSuperSecretTokenFound",
            ErrorMessage: "The request was not secret enough.",
        }).promise();
    } else {
        // If the user presented our custom 'SuperSecretToken' header, we send the
 requested object back to the user.
        // Again, note the presence of 'await'.
        const presignedResponse = await axios.get(inputS3Url);
```

```
        await s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            Body: presignedResponse.data,
        }).promise();
    }

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

Example 2: Respond with a transformed image

When performing an image transformation, you might find that you need all the bytes of the source object before you can start processing them. In this case, your `WriteGetObjectResponse` request returns the whole object to the requesting application in one call.

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.WriteGetObjectResponseRequest;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.awt.Image;
import java.io.ByteArrayOutputStream;
import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example2V2 {

    private static final int HEIGHT = 250;
    private static final int WIDTH = 250;
```

```
    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        S3Client s3Client = S3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Prepare the presigned URL for use and make the request to S3.
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // The entire image is loaded into memory here so that we can resize it.
        // Once the resizing is completed, we write the bytes into the body
        // of the WriteGetObjectResponse request.
        var originalImage = ImageIO.read(presignedResponse.body());
        var resizingImage = originalImage.getScaledInstance(WIDTH, HEIGHT,
 Image.SCALE_DEFAULT);
        var resizedImage = new BufferedImage(WIDTH, HEIGHT,
 BufferedImage.TYPE_INT_RGB);
        resizedImage.createGraphics().drawImage(resizingImage, 0, 0, WIDTH, HEIGHT,
 null);

        var baos = new ByteArrayOutputStream();
        ImageIO.write(resizedImage, "png", baos);

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(WriteGetObjectResponseRequest.builder()
                .requestRoute(event.outputRoute())
                .requestToken(event.outputToken())
                .build(), RequestBody.fromBytes(baos.toByteArray()));
    }
}
```

Python

```
import boto3
import requests
import io
from PIL import Image

def handler(event, context):
    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
```

```
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to
    S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    """
    In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
    'inputS3Url'.
    """
    image_request = requests.get(s3_url)
    image = Image.open(io.BytesIO(image_request.content))
    image.thumbnail((256,256), Image.ANTIALIAS)

    transformed = io.BytesIO()
    image.save(transformed, "png")

    # Send the resized image back to the client.
    s3 = boto3.client('s3')
    s3.write_get_object_response(Body=transformed.getvalue(), RequestRoute=route,
 RequestToken=token)

    # Gracefully exit the Lambda function.
    return { 'status_code': 200 }
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const sharp = require('sharp');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
```

```
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;

    // In this case, we're resizing .png images that are stored in S3 and are
 accessible through the presigned URL
    // 'inputS3Url'.
    const { data } = await axios.get(inputS3Url, { responseType: 'arraybuffer' });

    // Resize the image.
    const resized = await sharp(data)
        .resize({ width: 256, height: 256 })
        .toBuffer();

    // Send the resized image back to the client.
    await s3.writeGetObjectResponse({
        RequestRoute: outputRoute,
        RequestToken: outputToken,
        Body: resized,
    }).promise();

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

Example 3: Stream compressed content

When you're compressing objects, compressed data is produced incrementally. Consequently, you can use your `WriteGetObjectResponse` request to return the compressed data as soon as it's ready. As shown in this example, you don't need to know the length of the completed transformation.

Java

```
package com.amazon.s3.objectlambda;

import com.amazonaws.services.lambda.runtime.events.S3ObjectLambdaEvent;
import com.amazonaws.services.lambda.runtime.Context;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.WriteGetObjectResponseRequest;

import java.net.URI;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class Example3 {

    public void handleRequest(S3ObjectLambdaEvent event, Context context) throws
 Exception {
        S3Client s3Client = S3Client.builder().build();
        HttpClient httpClient = HttpClient.newBuilder().build();

        // Request the original object from S3.
        var presignedResponse = httpClient.send(
                HttpRequest.newBuilder(new URI(event.inputS3Url())).GET().build(),
                HttpResponse.BodyHandlers.ofInputStream());

        // Consume the incoming response body from the presigned request,
        // apply our transformation on that data, and emit the transformed bytes
        // into the body of the WriteGetObjectResponse request as soon as they're
 ready.
        // This example compresses the data from S3, but any processing pertinent
        // to your application can be performed here.
        var bodyStream = new GZIPCompressingInputStream(presignedResponse.body());

        // Stream the bytes back to the caller.
        s3Client.writeGetObjectResponse(WriteGetObjectResponseRequest.builder()
                .requestRoute(event.outputRoute())
                .requestToken(event.outputToken())
                .build(),
                RequestBody.fromInputStream(bodyStream,
                    presignedResponse.headers().firstValueAsLong("content-
length").orElse(-1L)));
    }
}
```

Python

```
import boto3
import requests
import zlib
```

```
from botocore.config import Config


"""
A helper class to work with content iterators. Takes an interator and compresses the
 bytes that come from it. It
implements 'read' and '__iter__' so that the SDK can stream the response.
"""
class Compress:
    def __init__(self, content_iter):
        self.content = content_iter
        self.compressed_obj = zlib.compressobj()

    def read(self, _size):
        for data in self.__iter__()
            return data

    def __iter__(self):
        while True:
            data = next(self.content)
            chunk = self.compressed_obj.compress(data)
            if not chunk:
                break

            yield chunk

        yield self.compressed_obj.flush()


def handler(event, context):
    """
    Setting the 'payload_signing_enabled' property to False allows us to send a
 streamed response back to the client.
    in this scenario, a streamed response means that the bytes are not buffered into
 memory as we're compressing them,
    but instead are sent straight to the user.
    """
    my_config = Config(
        region_name='eu-west-1',
        signature_version='s3v4',
        s3={
            "payload_signing_enabled": False
        }
    )
```

```
    s3 = boto3.client('s3', config=my_config)

    """
    Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    The 'userRequest' object has information related to the user who made this
 'GetObject' request to S3 Object Lambda.
    """
    get_context = event["getObjectContext"]
    route = get_context["outputRoute"]
    token = get_context["outputToken"]
    s3_url = get_context["inputS3Url"]

    # Compress the 'get' request stream.
    with requests.get(s3_url, stream=True) as r:
        compressed = Compress(r.iter_content())

        # Send the stream back to the client.
        s3.write_get_object_response(Body=compressed, RequestRoute=route,
 RequestToken=token, ContentType="text/plain",
                                     ContentEncoding="gzip")

    # Gracefully exit the Lambda function.
    return {'status_code': 200}
```

Node.js

```
const { S3 } = require('aws-sdk');
const axios = require('axios').default;
const zlib = require('zlib');

exports.handler = async (event) => {
    const s3 = new S3();

    // Retrieve the operation context object from the event. This object indicates
 where the WriteGetObjectResponse request
    // should be delivered and has a presigned URL in 'inputS3Url' where we can
 download the requested object from.
    const { getObjectContext } = event;
    const { outputRoute, outputToken, inputS3Url } = getObjectContext;
```

```
    // Download the object from S3 and process it as a stream, because it might be a
 huge object and we don't want to
    // buffer it in memory. Note the use of 'await' because we want to wait for
 'writeGetObjectResponse' to finish
    // before we can exit the Lambda function.
    await axios({
        method: 'GET',
        url: inputS3Url,
        responseType: 'stream',
    }).then(
        // Gzip the stream.
        response => response.data.pipe(zlib.createGzip())
    ).then(
        // Finally send the gzip-ed stream back to the client.
        stream => s3.writeGetObjectResponse({
            RequestRoute: outputRoute,
            RequestToken: outputToken,
            Body: stream,
            ContentType: "text/plain",
            ContentEncoding: "gzip",
        }).promise()
    );

    // Gracefully exit the Lambda function.
    return { statusCode: 200 };
}
```

Note

Although S3 Object Lambda allows up to 60 seconds to send a complete response to the caller through the `WriteGetObjectResponse` request, the actual amount of time available might be less. For example, your Lambda function timeout might be less than 60 seconds. In other cases, the caller might have more stringent timeouts.

For the original caller to receive a response other than HTTP status code 500 (Internal Server Error), the `WriteGetObjectResponse` call must be completed. If the Lambda function returns, with an exception or otherwise, before the `WriteGetObjectResponse` API operation is called, the original caller receives a 500 (Internal Server Error) response. Exceptions thrown during the time it takes to complete the response result in truncated responses to the caller. If the Lambda function

receives an HTTP status code 200 (OK) response from the `WriteGetObjectResponse` API call, then the original caller has sent the complete request. The Lambda function's response, whether an exception is thrown or not, is ignored by S3 Object Lambda.

When calling the `WriteGetObjectResponse` API operation, Amazon S3 requires the route and request token from the event context. For more information, see Event context format and usage.

The route and request token parameters are required to connect the `WriteGetObjectResult` response with the original caller. Even though it is always appropriate to retry 500 (Internal Server Error) responses, because the request token is a single-use token, subsequent attempts to use it might result in HTTP status code 400 (Bad Request) responses. Although the call to `WriteGetObjectResponse` with the route and request tokens doesn't need to be made from the invoked Lambda function, it must be made by an identity in the same account. The call also must be completed before the Lambda function finishes execution.

Working with `HeadObject` requests in Lambda

This section assumes that your Object Lambda Access Point is configured to call the Lambda function for `HeadObject`. Lambda will receive a JSON payload that contains a key called `headObjectContext`. Inside the context, there is a single property called `inputS3Url`, which is a presigned URL for the supporting access point for `HeadObject`.

The presigned URL will include the following properties if they're specified:

- `versionId` (in the query parameters)
- `requestPayer` (in the `x-amz-request-payer` header)
- `expectedBucketOwner` (in the `x-amz-expected-bucket-owner` header)

Other properties won't be presigned, and thus won't be included. Non-signed options sent as headers can be added manually to the request when calling the presigned URL that's found in the `userRequest` headers. Server-side encryption options are not supported for `HeadObject`.

For the request syntax URI parameters, see `HeadObject` in the *Amazon Simple Storage Service API Reference*.

The following example shows a Lambda JSON input payload for `HeadObject`.

```
{
  "xAmzRequestId": "requestId",
  "**headObjectContext**": {
```

```
    "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>"
  },
  "configuration": {
       "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
       "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
       "payload": "{}"
  },
  "userRequest": {
       "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
       "headers": {
           "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
           "Accept-Encoding": "identity",
           "X-Amz-Content-SHA256": "e3b0c44298fc1example"
       }
   },
   "userIdentity": {
       "type": "AssumedRole",
       "principalId": "principalId",
       "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
       "accountId": "111122223333",
       "accessKeyId": "accessKeyId",
       "sessionContext": {
            "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
       },
       "sessionIssuer": {
            "type": "Role",
            "principalId": "principalId",
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
            }
       }
    },
  "protocolVersion": "1.00"
}
```

Your Lambda function should return a JSON object that contains the headers and values that will be returned for the `HeadObject` call.

The following example shows the structure of the Lambda response JSON for `HeadObject`.

```
{
    "statusCode": <number>; // Required
    "errorCode": <string>;
    "errorMessage": <string>;
    "headers": {
        "Accept-Ranges": <string>,
        "x-amz-archive-status": <string>,
        "x-amz-server-side-encryption-bucket-key-enabled": <boolean>,
        "Cache-Control": <string>,
        "Content-Disposition": <string>,
        "Content-Encoding": <string>,
        "Content-Language": <string>,
        "Content-Length": <number>, // Required
        "Content-Type": <string>,
        "x-amz-delete-marker": <boolean>,
        "ETag": <string>,
        "Expires": <string>,
        "x-amz-expiration": <string>,
        "Last-Modified": <string>,
        "x-amz-missing-meta": <number>,
        "x-amz-object-lock-mode": <string>,
        "x-amz-object-lock-legal-hold": <string>,
        "x-amz-object-lock-retain-until-date": <string>,
        "x-amz-mp-parts-count": <number>,
        "x-amz-replication-status": <string>,
        "x-amz-request-charged": <string>,
        "x-amz-restore": <string>,
        "x-amz-server-side-encryption": <string>,
        "x-amz-server-side-encryption-customer-algorithm": <string>,
        "x-amz-server-side-encryption-aws-kms-key-id": <string>,
        "x-amz-server-side-encryption-customer-key-MD5": <string>,
        "x-amz-storage-class": <string>,
        "x-amz-tagging-count": <number>,
        "x-amz-version-id": <string>,
        <x-amz-meta-headers>: <string>, // user-defined metadata
        "x-amz-meta-meta1": <string>, // example of the user-defined metadata header,
 it will need the x-amz-meta prefix
        "x-amz-meta-meta2": <string>
        ...
```

```
    };
}
```

The following example shows how to use the presigned URL to populate your response by modifying the header values as needed before returning the JSON.

Python

```
import requests

def lambda_handler(event, context):
    print(event)

    # Extract the presigned URL from the input.
    s3_url = event["headObjectContext"]["inputS3Url"]

    # Get the head of the object from S3.
    response = requests.head(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        return {
            "statusCode": response.status_code,
            "errorCode": "RequestFailure",
            "errorMessage": "Request to S3 failed"
    }

    # Store the headers in a dictionary.
    response_headers = dict(response.headers)

    # This obscures Content-Type in a transformation, it is optional to add
    response_headers["Content-Type"] = ""

    # Return the headers to S3 Object Lambda.
    return {
        "statusCode": response.status_code,
        "headers": response_headers
        }
```

Working with `ListObjects` requests in Lambda

This section assumes that your Object Lambda Access Point is configured to call the Lambda function for `ListObjects`. Lambda will receive the JSON payload with a new object named `listObjectsContext`. `listObjectsContext`contains a single property, `inputS3Url`, which is a presigned URL for the supporting access point for `ListObjects`.

Unlike `GetObject` and `HeadObject`, the presigned URL will include the following properties if they're specified:

- All the query parameters
- `requestPayer` (in the `x-amz-request-payer` header)
- `expectedBucketOwner` (in the `x-amz-expected-bucket-owner` header)

For the request syntax URI parameters, see `ListObjects` in the *Amazon Simple Storage Service API Reference*.

> **Important**
>
> We recommend that you use the newer version, ListObjectsV2, when developing applications. For backward compatibility, Amazon S3 continues to support `ListObjects`.

The following example shows the Lambda JSON input payload for `ListObjects`.

```
{
    "xAmzRequestId": "requestId",
     "**listObjectsContext**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
```

```
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
  "protocolVersion": "1.00"
}
```

Your Lambda function should return a JSON object that contains the status code, list XML result, or error information that will be returned from S3 Object Lambda.

S3 Object Lambda does not process or validate `listResultXml`, but instead forwards it to `ListObjects` caller. For `listBucketResult`, S3 Object Lambda expects certain properties to be of a specific type and will throw exceptions if it cannot parse them. `listResultXml` and `listBucketResult` can not be provided at the same time.

The following example demonstrates how to use the presigned URL to call Amazon S3 and use the result to populate a response, including error checking.

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsContext"]["inputS3Url"]


    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""

    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)

    # Create response with listResultXml.
    response_with_list_result_xml = {
        'statusCode': 200,
        'listResultXml': listResultXml
    }

    # Create response with listBucketResult.
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
```

```
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
            newlist = []
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

The following example shows the structure of the Lambda response JSON for `ListObjects`.

```
{
  "statusCode": <number>; // Required
  "errorCode": <string>;
  "errorMessage": <string>;
  "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL

  "listBucketResult": {  // listBucketResult can be provided instead of listResultXml,
 however they can not both be provided in the JSON response
       "name": <string>,  // Required for 'listBucketResult'
       "prefix": <string>,
       "marker": <string>,
       "nextMarker": <string>,
       "maxKeys": <int>,   // Required for 'listBucketResult'
       "delimiter": <string>,
       "encodingType": <string>
       "isTruncated": <boolean>,  // Required for 'listBucketResult'
```

```
        "contents": [  {
            "key": <string>,  // Required for 'content'
            "lastModified": <string>,
            "eTag": <string>,
            "checksumAlgorithm": <string>,   // CRC32,  CRC32C,  SHA1,  SHA256
            "size": <int>,   // Required for 'content'
            "owner": {
                "displayName": <string>,  // Required for 'owner'
                "id": <string>,  // Required for 'owner'
            },
            "storageClass": <string>
            },
        ...
        ],
        "commonPrefixes": [  {
            "prefix": <string>   // Required for 'commonPrefix'
        },
        ...
        ],
    }
}
```

Working with `ListObjectsV2` requests in Lambda

This section assumes that your Object Lambda Access Point is configured to call the Lambda function for `ListObjectsV2`. Lambda will receive the JSON payload with a new object named `listObjectsV2Context`. `listObjectsV2Context` contains a single property, `inputS3Url`, which is a presigned URL for the supporting access point for `ListObjectsV2`.

Unlike `GetObject` and `HeadObject`, the presigned URL will include the following properties, if they're specified:

- All the query parameters
- `requestPayer` (in the `x-amz-request-payer` header)
- `expectedBucketOwner` (in the `x-amz-expected-bucket-owner` header)

For the request syntax URI parameters, see `ListObjectsV2` in the *Amazon Simple Storage Service API Reference*.

The following example shows the Lambda JSON input payload for `ListObjectsV2`.

```
{
    "xAmzRequestId": "requestId",
     "**listObjectsV2Context**": {
     "**inputS3Url**": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/?list-type=2&X-Amz-Security-Token=<snip>",
     },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
```

```
  "protocolVersion": "1.00"
}
```

Your Lambda function should return a JSON object that contains the status code, list XML result, or error information that will be returned from S3 Object Lambda.

S3 Object Lambda does not process or validate `listResultXml`, but instead forwards it to `ListObjectsV2` caller. For `listBucketResult`, S3 Object Lambda expects certain properties to be of a specific type and will throw exceptions if it cannot parse them. `listResultXml` and `listBucketResult` can not be provided at the same time.

The following example demonstrates how to use the presigned URL to call Amazon S3 and use the result to populate a response, including error checking.

Python

```
import requests
import xmltodict

def lambda_handler(event, context):
    # Extract the presigned URL from the input.
    s3_url = event["listObjectsV2Context"]["inputS3Url"]


    # Get the head of the object from Amazon S3.
    response = requests.get(s3_url)

    # Return the error to S3 Object Lambda (if applicable).
    if (response.status_code >= 400):
        error = xmltodict.parse(response.content)
        return {
            "statusCode": response.status_code,
            "errorCode": error["Error"]["Code"],
            "errorMessage": error["Error"]["Message"]
        }

    # Store the XML result in a dict.
    response_dict = xmltodict.parse(response.content)

    # This obscures StorageClass in a transformation, it is optional to add
    for item in response_dict['ListBucketResult']['Contents']:
        item['StorageClass'] = ""
```

```
    # Convert back to XML.
    listResultXml = xmltodict.unparse(response_dict)

    # Create response with listResultXml.
    response_with_list_result_xml = {
        'statusCode': 200,
        'listResultXml': listResultXml
    }

    # Create response with listBucketResult.
    response_dict['ListBucketResult'] =
 sanitize_response_dict(response_dict['ListBucketResult'])
    response_with_list_bucket_result = {
        'statusCode': 200,
        'listBucketResult': response_dict['ListBucketResult']
    }

    # Return the list to S3 Object Lambda.
    # Can return response_with_list_result_xml or response_with_list_bucket_result
    return response_with_list_result_xml

# Converting the response_dict's key to correct casing
def sanitize_response_dict(response_dict: dict):
    new_response_dict = dict()
    for key, value in response_dict.items():
        new_key = key[0].lower() + key[1:] if key != "ID" else 'id'
        if type(value) == list:
            newlist = []
            for element in value:
                if type(element) == type(dict()):
                    element = sanitize_response_dict(element)
                newlist.append(element)
            value = newlist
        elif type(value) == dict:
            value = sanitize_response_dict(value)
        new_response_dict[new_key] = value
    return new_response_dict
```

The following example shows the structure of the Lambda response JSON for `ListObjectsV2`.

```
{
    "statusCode": <number>; // Required
```

```
    "errorCode": <string>;
    "errorMessage": <string>;
    "listResultXml": <string>; // This can also be Error XML string in case S3 returned
 error response when calling the pre-signed URL

    "listBucketResult": {  // listBucketResult can be provided instead of
 listResultXml, however they can not both be provided in the JSON response
        "name": <string>, // Required for 'listBucketResult'
        "prefix": <string>,
        "startAfter": <string>,
        "continuationToken": <string>,
        "nextContinuationToken": <string>,
        "keyCount": <int>, // Required for 'listBucketResult'
        "maxKeys": <int>, // Required for 'listBucketResult'
        "delimiter": <string>,
        "encodingType": <string>
        "isTruncated": <boolean>, // Required for 'listBucketResult'
        "contents": [ {
            "key": <string>, // Required for 'content'
            "lastModified": <string>,
            "eTag": <string>,
            "checksumAlgorithm": <string>, // CRC32, CRC32C, SHA1, SHA256
            "size": <int>, // Required for 'content'
            "owner": {
                "displayName": <string>, // Required for 'owner'
                "id": <string>, // Required for 'owner'
            },
            "storageClass": <string>
            },
            ...
        ],
        "commonPrefixes": [ {
            "prefix": <string> // Required for 'commonPrefix'
            },
        ...
        ],
    }
}
```

Event context format and usage

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

Amazon S3 Object Lambda provides context about the request that's being made in the event that's passed to your AWS Lambda function. The following shows an example request. Descriptions of the fields are included after the example.

```
{
    "xAmzRequestId": "requestId",
    "getObjectContext": {
        "inputS3Url": "https://my-s3-ap-111122223333.s3-accesspoint.us-
east-1.amazonaws.com/example?X-Amz-Security-Token=<snip>",
        "outputRoute": "io-use1-001",
        "outputToken": "OutputToken"
    },
    "configuration": {
        "accessPointArn": "arn:aws:s3-object-lambda:us-
east-1:111122223333:accesspoint/example-object-lambda-ap",
        "supportingAccessPointArn": "arn:aws:s3:us-
east-1:111122223333:accesspoint/example-ap",
        "payload": "{}"
    },
    "userRequest": {
        "url": "https://object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com/example",
        "headers": {
            "Host": "object-lambda-111122223333.s3-object-lambda.us-
east-1.amazonaws.com",
            "Accept-Encoding": "identity",
            "X-Amz-Content-SHA256": "e3b0c44298fc1example"
        }
    },
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "principalId",
```

```
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/example",
        "accountId": "111122223333",
        "accessKeyId": "accessKeyId",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "Wed Mar 10 23:41:52 UTC 2021"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "principalId",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            }
        }
    },
    "protocolVersion": "1.00"
}
```

The following fields are included in the request:

- `xAmzRequestId` – The Amazon S3 request ID for this request. We recommend that you log this value to help with debugging.
- `getObjectContext` – The input and output details for connections to Amazon S3 and S3 Object Lambda.
  - `inputS3Url` – A presigned URL that can be used to fetch the original object from Amazon S3. The URL is signed by using the original caller's identity, and that user's permissions will apply when the URL is used. If there are signed headers in the URL, the Lambda function must include these headers in the call to Amazon S3, except for the `Host` header.
  - `outputRoute` – A routing token that is added to the S3 Object Lambda URL when the Lambda function calls `WriteGetObjectResponse`.
  - `outputToken` – An opaque token that's used by S3 Object Lambda to match the `WriteGetObjectResponse` call with the original caller.
- `configuration` – Configuration information about the Object Lambda Access Point.
  - `accessPointArn` – The Amazon Resource Name (ARN) of the Object Lambda Access Point that received this request.

  - `supportingAccessPointArn` – The ARN of the supporting access point that is specified in the Object Lambda Access Point configuration.
  - `payload` – Custom data that is applied to the Object Lambda Access Point configuration. S3 Object Lambda treats this data as an opaque string, so it might need to be decoded before use.
- `userRequest` – Information about the original call to S3 Object Lambda.
  - `url` – The decoded URL of the request as received by S3 Object Lambda, excluding any authorization-related query parameters.
  - `headers` – A map of string to strings containing the HTTP headers and their values from the original call, excluding any authorization-related headers. If the same header appears multiple times, the values from each instance of the same header are combined into a comma-delimited list. The case of the original headers is retained in this map.
- `userIdentity` – Details about the identity that made the call to S3 Object Lambda. For more information, see Logging data events for trails in the *AWS CloudTrail User Guide*.
  - `type` – The type of identity.
  - `accountId` – The AWS account to which the identity belongs.
  - `userName` – The friendly name of the identity that made the call.
  - `principalId` – The unique identifier for the identity that made the call.
  - `arn` – The ARN of the principal who made the call. The last section of the ARN contains the user or role that made the call.
  - `sessionContext` – If the request was made with temporary security credentials, this element provides information about the session that was created for those credentials.
  - `invokedBy` – The name of the AWS service that made the request, such as Amazon EC2 Auto Scaling or AWS Elastic Beanstalk.
  - `sessionIssuer` – If the request was made with temporary security credentials, this element provides information about how the credentials were obtained.
- `protocolVersion` – The version ID of the context provided. The format of this field is `{Major Version}.{Minor Version}`. The minor version numbers are always two-digit numbers. Any removal or change to the semantics of a field necessitates a major version bump and requires active opt-in. Amazon S3 can add new fields at any time, at which point you might experience a minor version bump. Because of the nature of software rollouts, you might see multiple minor versions in use at once.

Working with Range and partNumber headers

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

When working with large objects in Amazon S3 Object Lambda, you can use the `Range` HTTP header to download a specified byte range from an object. To fetch different byte ranges from within the same object, you can use concurrent connections to Amazon S3. You can also specify the `partNumber` parameter (an integer between 1 and 10,000), which performs a ranged request for the specified part of the object.

Because there are multiple ways that you might want to handle a request that includes the `Range` or `partNumber` parameters, S3 Object Lambda doesn't apply these parameters to the transformed object. Instead, your AWS Lambda function must implement this functionality as needed for your application.

To use the `Range` and `partNumber` parameters with S3 Object Lambda, you do the following:

- Enable these parameters in your Object Lambda Access Point configuration.
- Write a Lambda function that can handle requests that include these parameters.

The following steps describe how to accomplish this.

Step 1: Configure your Object Lambda Access Point

By default, Object Lambda Access Points respond with an HTTP status code 501 (Not Implemented) error to any `GetObject` or `HeadObject` request that contains a `Range` or `partNumber` parameter, either in the headers or query parameters.

To enable an Object Lambda Access Point to accept such requests, you must include `GetObject-Range`, `GetObject-PartNumber`, `HeadObject-Range`, or `HeadObject-PartNumber` in the `AllowedFeatures` section of your Object Lambda Access Point configuration. For more information about updating your Object Lambda Access Point configuration, see Creating Object Lambda Access Points.

Step 2: Implement Range or `partNumber` handling in your Lambda function

When your Object Lambda Access Point invokes your Lambda function with a ranged `GetObject` or `HeadObject` request, the `Range` or `partNumber` parameter is included in the event context. The location of the parameter in the event context depends on which parameter was used and how it was included in the original request to the Object Lambda Access Point, as explained in the following table.

Parameter	Event context location
`Range` (header)	`userRequest.headers.Range`
`Range` (query parameter)	`userRequest.url` (query parameter `Range`)
`partNumber`	`userRequest.url` (query parameter `partNumber`)

> **Important**
>
> The provided presigned URL for your Object Lambda Access Point doesn't contain the `Range` or `partNumber` parameter from the original request. See the following options on how to handle these parameters in your AWS Lambda function.

After you extract the `Range` or `partNumber` value, you can take one of the following approaches, based on your application's needs:

A. **Map the requested `Range` or `partNumber` to the transformed object (recommended).**

The most reliable way to handle `Range` or `partNumber` requests is to do the following:

- Retrieve the full object from Amazon S3.
- Transform the object.
- Apply the requested `Range` or `partNumber` parameters to the transformed object.

To do this, use the provided presigned URL to fetch the entire object from Amazon S3 and then process the object as needed. For an example Lambda function that processes a `Range` parameter in this way, see this sample in the AWS Samples GitHub repository.

B. **Map the requested Range to the presigned URL.**

In some cases, your Lambda function can map the requested Range directly to the presigned URL to retrieve only part of the object from Amazon S3. This approach is appropriate only if your transformation meets both of the following criteria:

1. Your transformation function can be applied to partial object ranges.
2. Applying the Range parameter before or after the transformation function results in the same transformed object.

For example, a transformation function that converts all characters in an ASCII-encoded object to uppercase meets both of the preceding criteria. The transformation can be applied to part of an object, and applying the Range parameter before the transformation achieves the same result as applying it after the transformation.

By contrast, a function that reverses the characters in an ASCII-encoded object doesn't meet these criteria. Such a function meets criterion 1, because it can be applied to partial object ranges. However, it doesn't meet criterion 2, because applying the Range parameter before the transformation achieves different results than applying the parameter after the transformation.

Consider a request to apply the function to the first three characters of an object with the contents `abcdefg`. Applying the Range parameter before the transformation retrieves only `abc` and then reverses the data, returning `cba`. But if the parameter is applied after the transformation, the function retrieves the entire object, reverses it, and then applies the Range parameter, returning `gfe`. Because these results are different, this function should not apply the Range parameter when retrieving the object from Amazon S3. Instead, it should retrieve the entire object, perform the transformation, and only then apply the Range parameter.

> **Warning**
>
> In many cases, applying the Range parameter to the presigned URL will result in unexpected behavior by the Lambda function or the requesting client. Unless you are sure that your application will work properly when retrieving only a partial object from Amazon S3, we recommend that you retrieve and transform full objects as described earlier in approach A.

If your application meets the criteria described earlier in approach B, you can simplify your AWS Lambda function by fetching only the requested object range and then running your transformation on that range.

The following Java code example demonstrates how to do the following:

- Retrieve the `Range` header from the `GetObject` request.
- Add the `Range` header to the presigned URL that Lambda can use to retrieve the requested range from Amazon S3.

```
private HttpRequest.Builder applyRangeHeader(ObjectLambdaEvent event,
 HttpRequest.Builder presignedRequest) {
    var header = event.getUserRequest().getHeaders().entrySet().stream()
            .filter(e -> e.getKey().toLowerCase(Locale.ROOT).equals("range"))
            .findFirst();

    // Add check in the query string itself.
    header.ifPresent(entry -> presignedRequest.header(entry.getKey(),
 entry.getValue()));
    return presignedRequest;
}
```

Using AWS built Lambda functions

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

AWS provides some prebuilt AWS Lambda functions that you can use with Amazon S3 Object Lambda to detect and redact personally identifiable information (PII) and decompress S3 objects. These Lambda functions are available in the AWS Serverless Application Repository. You can select these functions through the AWS Management Console when you create your Object Lambda Access Point.

For more information about how to deploy serverless applications from the AWS Serverless Application Repository, see Deploying Applications in the *AWS Serverless Application Repository Developer Guide*.

> **Note**
>
> The following examples can be used only with `GetObject` requests.

Example 1: PII access control

This Lambda function uses Amazon Comprehend, a natural language processing (NLP) service that uses machine learning to find insights and relationships in text. This function automatically detects personally identifiable information (PII), such as names, addresses, dates, credit card numbers, and social security numbers in documents in your Amazon S3 bucket. If you have documents in your bucket that include PII, you can configure the PII Access Control function to detect these PII entity types and restrict access to unauthorized users.

To get started, deploy the following Lambda function in your account and add the Amazon Resource Name (ARN) for the function to your Object Lambda Access Point configuration.

The following is an example ARN for this function:

```
arn:aws:serverlessrepo:us-east-1:111122223333:applications/
ComprehendPiiAccessControlS3ObjectLambda
```

You can add or view this function on the AWS Management Console by using the following AWS Serverless Application Repository link: ComprehendPiiAccessControlS3ObjectLambda.

To view this function on GitHub, see Amazon Comprehend S3 Object Lambda.

Example 2: PII redaction

This Lambda function uses Amazon Comprehend, a natural language processing (NLP) service that uses machine learning to find insights and relationships in text. This function automatically redacts personally identifiable information (PII), such as names, addresses, dates, credit card numbers, and social security numbers from documents in your Amazon S3 bucket.

If you have documents in your bucket that include information such as credit card numbers or bank account information, you can configure the PII Redaction S3 Object Lambda function to detect PII and then return a copy of these documents in which PII entity types are redacted.

To get started, deploy the following Lambda function in your account and add the ARN for the function to your Object Lambda Access Point configuration.

The following is an example ARN for this function:

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/
ComprehendPiiRedactionS3ObjectLambda
```

You can add or view this function on the AWS Management Console by using the following AWS Serverless Application Repository link: ComprehendPiiRedactionS3ObjectLambda.

To view this function on GitHub, see Amazon Comprehend S3 Object Lambda.

To learn about complete end-to-end procedures for some S3 Object Lambda tasks in PII redaction, see Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend.

Example 3: Decompression

The Lambda function `S3ObjectLambdaDecompression` can decompress objects that are stored in Amazon S3 in one of six compressed file formats: `bzip2`, `gzip`, `snappy`, `zlib`, `zstandard`, and `ZIP`.

To get started, deploy the following Lambda function in your account and add the ARN for the function to your Object Lambda Access Point configuration.

The following is an example ARN for this function:

```
arn:aws:serverlessrepo:us-east-1:111122223333::applications/S3ObjectLambdaDecompression
```

You can add or view this function on the AWS Management Console by using the following AWS Serverless Application Repository link: S3ObjectLambdaDecompression.

To view this function on GitHub, see S3 Object Lambda Decompression.

Best practices and guidelines for S3 Object Lambda

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

When using S3 Object Lambda, follow these best practices and guidelines to optimize operations and performance.

Topics

- Working with S3 Object Lambda
- AWS services used in connection with S3 Object Lambda
- Range and partNumber headers
- Transforming the expiry-date
- Working with the AWS CLI and AWS SDKs

Working with S3 Object Lambda

S3 Object Lambda supports processing only `GET`, `LIST`, and `HEAD` requests. Any other requests don't invoke AWS Lambda and instead return standard, non-transformed API responses. You can create a maximum of 1,000 Object Lambda Access Points per AWS account per Region. The AWS Lambda function that you use must be in the same AWS account and Region as the Object Lambda Access Point.

S3 Object Lambda allows up to 60 seconds to stream a complete response to its caller. Your function is also subject to AWS Lambda default quotas. For more information, see Lambda quotas in the *AWS Lambda Developer Guide*.

When S3 Object Lambda invokes your specified Lambda function, you are responsible for ensuring that any data that is overwritten or deleted from Amazon S3 by your specified Lambda function or application is intended and correct.

You can use S3 Object Lambda only to perform operations on objects. You cannot use S3 Object Lambda to perform other Amazon S3 operations, such as modifying or deleting buckets. For a complete list of S3 operations that support access points, see Access points compatibility with S3 operations.

In addition to this list, Object Lambda Access Points do not support the `POST Object`, `CopyObject` (as the source), and `SelectObjectContent` API operations.

AWS services used in connection with S3 Object Lambda

S3 Object Lambda connects Amazon S3, AWS Lambda, and optionally, other AWS services of your choosing to deliver objects relevant to the requesting applications. All AWS services used with S3 Object Lambda are governed by their respective Service Level Agreements (SLAs). For example, if any AWS service does not meet its Service Commitment, you are eligible to receive a Service Credit, as documented in the service's SLA.

Range and `partNumber` headers

When working with large objects, you can use the `Range` HTTP header to download a specified byte-range from an object. When you use the `Range` header, your request fetches only the specified portion of the object. You can also use the `partNumber` header to perform a ranged request for the specified part from the object.

For more information, see Working with Range and partNumber headers.

Transforming the `expiry-date`

You can open or download transformed objects from your Object Lambda Access Point on the AWS Management Console. These objects must be non-expired. If your Lambda function transforms the `expiry-date` of your objects, you might see expired objects that cannot be opened or downloaded. This behavior applies only to S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive restored objects.

Working with the AWS CLI and AWS SDKs

AWS Command Line Interface (AWS CLI) S3 subcommands (`cp`, `mv`, and `sync`) and the use of the AWS SDK for Java `TransferManager` class are not supported for use with S3 Object Lambda.

S3 Object Lambda tutorials

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

The following tutorials present complete end-to-end procedures for some S3 Object Lambda tasks.

With S3 Object Lambda, you can add your own code to process data retrieved from S3 before returning it to an application. Each of the following tutorials will modify data as it is retrieved from Amazon S3, without changing the existing object or maintaining multiple copies of the data. The first tutorial will walk through how to add an AWS Lambda function to a S3 GET request to modify an object retrieved from S3. The second tutorial demonstrates how to use a prebuilt Lambda function powered by Amazon Comprehend to protect personally identifiable information (PII) retrieved from S3 before returning it to an application. The third tutorial uses S3 Object Lambda to add a watermark to an image as it is retrieved from Amazon S3.

- Tutorial: Transforming data for your application with S3 Object Lambda
- Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved

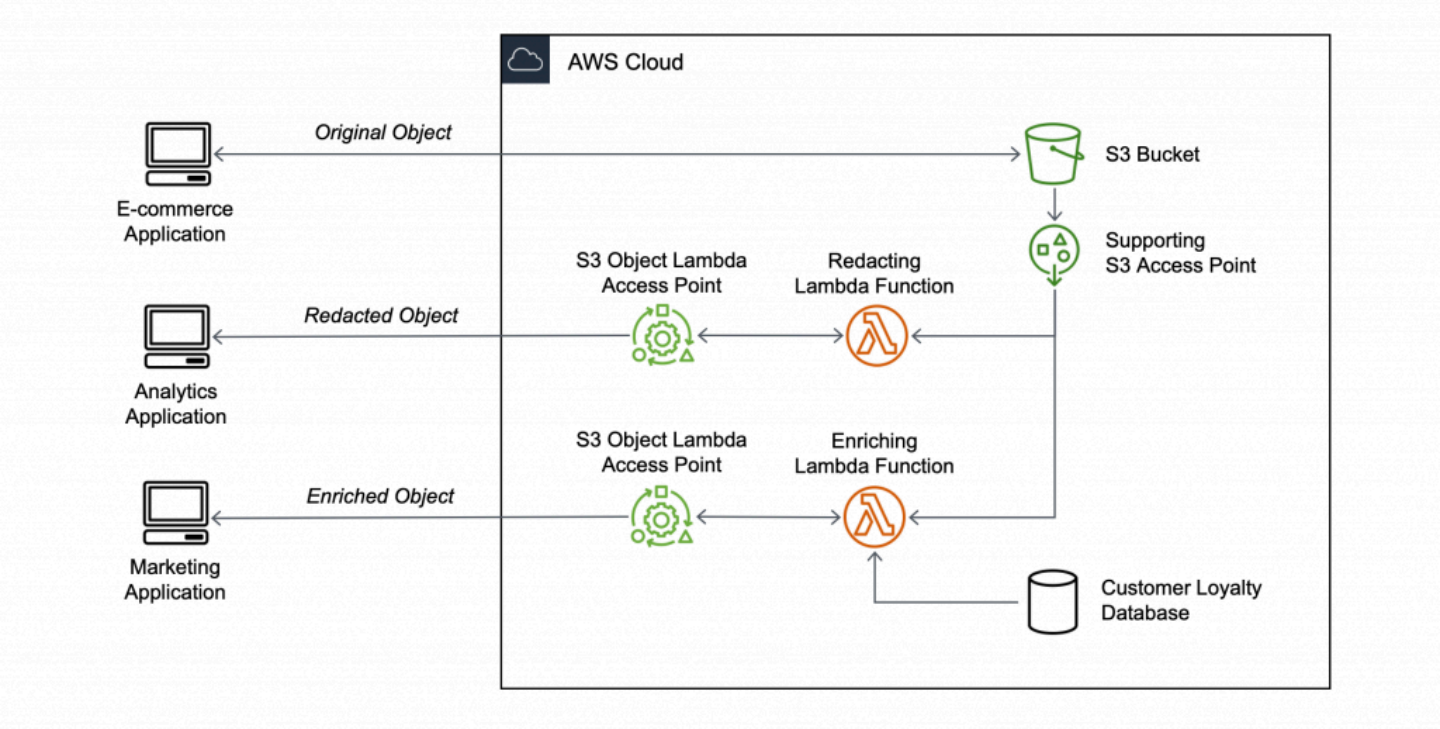
Tutorial: Transforming data for your application with S3 Object Lambda

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

When you store data in Amazon S3, you can easily share it for use by multiple applications. However, each application might have unique data format requirements, and might need

modification or processing of your data for a specific use case. For example, a dataset created by an ecommerce application might include personally identifiable information (PII). When the same data is processed for analytics, this PII is not needed and should be redacted. However, if the same dataset is used for a marketing campaign, you might need to enrich the data with additional details, such as information from the customer loyalty database.

With S3 Object Lambda, you can add your own code to process data retrieved from S3 before returning it to an application. Specifically, you can configure an AWS Lambda function and attach it to an S3 Object Lambda Access Point. When an application sends standard S3 GET requests through the S3 Object Lambda Access Point, the specified Lambda function is invoked to process any data retrieved from the underlying data source through the supporting S3 access point. Then, the S3 Object Lambda Access Point returns the transformed result back to the application. You can author and execute your own custom Lambda functions, tailoring the S3 Object Lambda data transformation to your specific use case, all with no changes required to your applications.



Objective

In this tutorial, you learn how to add custom code to standard S3 GET requests to modify the requested object retrieved from S3 so that the object suit the needs of the requesting client or application. Specifically, you learn how to transform all the text in the original object stored in an S3 bucket to uppercase through S3 Object Lambda.

> **Note**
>
> This tutorial uses Python code to transform the data, for examples using other AWS SDKs see Transform data for your application with S3 Object Lambda in the AWS SDK Code Examples Library.

Topics

- Prerequisites
- Step 1: Create an S3 bucket
- Step 2: Upload a file to the S3 bucket
- Step 3: Create an S3 access point
- Step 4: Create a Lambda function
- Step 5: Configure an IAM policy for your Lambda function's execution role
- Step 6: Create an S3 Object Lambda Access Point
- Step 7: View the transformed data
- Step 8: Clean up
- Next steps

Prerequisites

Before you start this tutorial, you must have an AWS account that you can sign in to as an AWS Identity and Access Management (IAM) user with correct permissions. You also must install Python version 3.8 or later.

Substeps

- Create an IAM user with permissions in your AWS account (console)
- Install Python 3.8 or later on your local machine

Create an IAM user with permissions in your AWS account (console)

You can create an IAM user for the tutorial. To complete this tutorial, your IAM user must attach the following IAM policies to access relevant AWS resources and perform specific actions. For more

information about how to create an IAM user, see Creating IAM users (console) in the *IAM User Guide*.

Your IAM user requires the following policies:

- AmazonS3FullAccess – Grants permissions to all Amazon S3 actions, including permissions to create and use an Object Lambda Access Point.
- AWSLambda_FullAccess – Grants permissions to all Lambda actions.
- IAMFullAccess – Grants permissions to all IAM actions.
- IAMAccessAnalyzerReadOnlyAccess – Grants permissions to read all access information provided by IAM Access Analyzer.
- CloudWatchLogsFullAccess – Grants full access to CloudWatch Logs.

> **Note**
>
> For simplicity, this tutorial creates and uses an IAM user. After completing this tutorial, remember to Delete the IAM user. For production use, we recommend that you follow the Security best practices in IAM in the *IAM User Guide*. A best practice requires human users to use federation with an identity provider to access AWS with temporary credentials. Another best practice is to require workloads to use temporary credentials with IAM roles to access AWS. To learn about using AWS IAM Identity Center to create users with temporary credentials, see Getting started in the *AWS IAM Identity Center User Guide*. This tutorial also uses full-access AWS managed policies. For production use, we recommend that you instead grant only the minimum permissions necessary for your use case, in accordance with security best practices.

Install Python 3.8 or later on your local machine

Use the following procedure to install Python 3.8 or later on your local machine. For more installation instructions, see the Downloading Python page in the *Python Beginners Guide*.

1. Open your local terminal or shell and run the following command to determine whether Python is already installed, and if so, which version is installed.

```
python --version
```

2. If you don't have Python 3.8 or later, download the official installer of Python 3.8 or later that's suitable for your local machine.
3. Run the installer by double-clicking the downloaded file, and follow the steps to complete the installation.

   For **Windows users**, choose **Add Python 3.X to PATH** in the installation wizard before choosing **Install Now**.
4. Restart your terminal by closing and reopening it.
5. Run the following command to verify that Python 3.8 or later is installed correctly.

   For **macOS users**, run this command:

   ```
   python3 --version
   ```

   For **Windows users**, run this command:

   ```
   python --version
   ```
6. Run the following command to verify that the pip3 package manager is installed. If you see a pip version number and python 3.8 or later in the command response, that means the pip3 package manager is installed successfully.

   ```
   pip --version
   ```

Step 1: Create an S3 bucket

Create a bucket to store the original data that you plan to transform.

> **Note**
>
> Access points may be attached for another data source, such as an Amazon FSx for OpenZFS volume, however this tutorial uses a supporting access point attached to an S3 bucket.

To create a bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.

   The **Create bucket** page opens.
4. For **Bucket name**, enter a name (for example, **`tutorial-bucket`**) for your bucket.

   For more information about naming buckets in Amazon S3, see General purpose bucket naming rules.
5. For **Region**, choose the AWS Region where you want the bucket to reside.

   For more information about the bucket Region, see General purpose buckets overview.
6. For **Block Public Access settings for this bucket**, keep the default settings (**Block *all* public access** is enabled).

   We recommend that you keep all Block Public Access settings enabled unless you need to turn off one or more of them for your use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.
7. For the remaining settings, keep the defaults.

   (Optional) If you want to configure additional bucket settings for your specific use case, see Creating a general purpose bucket.
8. Choose **Create bucket**.

Step 2: Upload a file to the S3 bucket

Upload a text file to the S3 bucket. This text file contains the original data that you will transform to uppercase later in this tutorial.

For example, you can upload a `tutorial.txt` file that contains the following text:

```
Amazon S3 Object Lambda Tutorial:
You can add your own code to process data retrieved from S3 before
returning it to an application.
```

To upload a file to a bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**) to upload your file to.
4. On the **Objects** tab for your bucket, choose **Upload**.
5. On the **Upload** page, under **Files and folders**, choose **Add files**.
6. Choose a file to upload, and then choose **Open**. For example, you can upload the `tutorial.txt` file example mentioned earlier.
7. Choose **Upload**.

Step 3: Create an S3 access point

To use an S3 Object Lambda Access Point to access and transform the original data, you must create an S3 access point and associate it with the S3 bucket that you created in Step 1. The access point must be in the same AWS Region as the objects that you want to transform.

Later in this tutorial, you'll use this access point as a supporting access point for your Object Lambda Access Point.

To create an access point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points**.
3. On the **Access Points** page, choose **Create access point**.
4. In the **Access point name** field, enter the name (for example, **`tutorial-access-point`**) for the access point.

   For more information about naming access points, see Naming rules for access points.
5. In the **Data source** field, enter the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**). S3 attaches the access point to this bucket.

(Optional) You can choose **Browse S3** to browse and search the buckets in your account. If you choose **Browse S3**, choose the desired bucket, and then choose **Choose path** to populate the **Bucket name** field with that bucket's name.

6. For **Network origin**, choose **Internet**.

   For more information about network origins for access points, see Creating access points restricted to a virtual private cloud.

7. By default, all Block Public Access settings are turned on for your access point. We recommend that you keep **Block *all* public access** enabled.

   For more information, see Managing public access to access points for general purpose buckets.

8. For all other access point settings, keep the default settings.

   (Optional) You can modify the access point settings to support your use case. For this tutorial, we recommend keeping the default settings.

   (Optional) If you need to manage access to your access point, you can specify an access point policy. For more information, see Policy examples for access points.

9. Choose **Create access point**.

Step 4: Create a Lambda function

To transform original data, create a Lambda function for use with your S3 Object Lambda Access Point.

Substeps

- Write Lambda function code and create a deployment package with a virtual environment
- Create a Lambda function with an execution role (console)
- Deploy your Lambda function code with .zip file archives and configure the Lambda function (console)

Write Lambda function code and create a deployment package with a virtual environment

1. On your local machine, create a folder with the folder name `object-lambda` for the virtual environment to use later in this tutorial.

2. In the `object-lambda` folder, create a file with a Lambda function that changes all text in the original object to uppercase. For example, you can use the following function written in Python. Save this function in a file named `transform.py`.

```
import boto3
import requests
from botocore.config import Config

# This function capitalizes all text in the original object
def lambda_handler(event, context):
    object_context = event["getObjectContext"]
    # Get the presigned URL to fetch the requested original object
    # from S3
    s3_url = object_context["inputS3Url"]
    # Extract the route and request token from the input context
    request_route = object_context["outputRoute"]
    request_token = object_context["outputToken"]

    # Get the original S3 object using the presigned URL
    response = requests.get(s3_url)
    original_object = response.content.decode("utf-8")

    # Transform all text in the original object to uppercase
    # You can replace it with your custom code based on your use case
    transformed_object = original_object.upper()

    # Write object back to S3 Object Lambda
    s3 = boto3.client('s3', config=Config(signature_version='s3v4'))
    # The WriteGetObjectResponse API sends the transformed data
    # back to S3 Object Lambda and then to the user
    s3.write_get_object_response(
        Body=transformed_object,
        RequestRoute=request_route,
        RequestToken=request_token)

    # Exit the Lambda function: return the status code
    return {'status_code': 200}
```

Note

The preceding example Lambda function loads the entire requested object into memory before transforming it and returning it to the client. Alternatively, you

can stream the object from S3 to avoid loading the entire object into memory. This approach can be useful when working with large objects. For more information about streaming responses with Object Lambda Access Points, see the streaming examples in Working with `GetObject` requests in Lambda.

When you're writing a Lambda function for use with an S3 Object Lambda Access Point, the function is based on the input event context that S3 Object Lambda provides to the Lambda function. The event context provides information about the request being made in the event passed from S3 Object Lambda to Lambda. It contains the parameters that you use to create the Lambda function.

The fields used to create the preceding Lambda function are as follows:

The field of `getObjectContext` means the input and output details for connections to Amazon S3 and S3 Object Lambda. It has the following fields:

- `inputS3Url` – A presigned URL that the Lambda function can use to download the original object from the supporting access point. By using a presigned URL, the Lambda function doesn't need to have Amazon S3 read permissions to retrieve the original object and can only access the object processed by each invocation.
- `outputRoute` – A routing token that is added to the S3 Object Lambda URL when the Lambda function calls `WriteGetObjectResponse` to send back the transformed object.
- `outputToken` – A token used by S3 Object Lambda to match the `WriteGetObjectResponse` call with the original caller when sending back the transformed object.

For more information about all the fields in the event context, see Event context format and usage and Writing Lambda functions for S3 Object Lambda Access Points.

3. In your local terminal, enter the following command to install the `virtualenv` package:

```
python -m pip install virtualenv
```

4. In your local terminal, open the `object-lambda` folder that you created earlier, and then enter the following command to create and initialize a virtual environment called `venv`.

```
python -m virtualenv venv
```

5. To activate the virtual environment, enter the following command to execute the `activate` file from the environment's folder:

   For **macOS users**, run this command:

   ```
   source venv/bin/activate
   ```

   For **Windows users**, run this command:

   ```
   .\venv\Scripts\activate
   ```

   Now, your command prompt changes to show **(venv)**, indicating that the virtual environment is active.

6. To install the required libraries, run the following commands line by line in the `venv` virtual environment.

   These commands install updated versions of the dependencies of your `lambda_handler` Lambda function. These dependencies are the AWS SDK for Python (Boto3) and the requests module.

   ```
   pip3 install boto3
   ```

   ```
   pip3 install requests
   ```

7. To deactivate the virtual environment, run the following command:

   ```
   deactivate
   ```

8. To create a deployment package with the installed libraries as a `.zip` file named `lambda.zip` at the root of the `object-lambda` directory, run the following commands line by line in your local terminal.

   Tip

   The following commands might need to be adjusted to work in your particular environment. For example, a library might appear in `site-packages` or `dist-packages`, and the first folder might be `lib` or `lib64`. Also, the `python` folder might

be named with a different Python version. To locate a specific package, use the `pip show` command.

For **macOS users**, run these commands:

```
cd venv/lib/python3.8/site-packages
```

```
zip -r ../../../../lambda.zip .
```

For **Windows users**, run these commands:

```
cd .\venv\Lib\site-packages\
```

```
powershell Compress-Archive * ../../../lambda.zip
```

The last command saves the deployment package to the root of the `object-lambda` directory.

9. Add the function code file `transform.py` to the root of your deployment package.

For **macOS users**, run these commands:

```
cd ../../../../
```

```
zip -g lambda.zip transform.py
```

For **Windows users**, run these commands:

```
cd ..\..\..\
```

```
powershell Compress-Archive -update transform.py lambda.zip
```

After you complete this step, you should have the following directory structure:

```
lambda.zip$
  # transform.py
```

```
# __pycache__
| boto3/
# certifi/
# pip/
# requests/
...
```

Create a Lambda function with an execution role (console)

1. Sign in to the AWS Management Console and open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. In the left navigation pane, choose **Functions**.
3. Choose **Create function**.
4. Choose **Author from scratch**.
5. Under **Basic information**, do the following:
   a. For **Function name**, enter **`tutorial-object-lambda-function`**.
   b. For **Runtime**, choose **Python 3.8** or a later version.
6. Expand the **Change default execution role** section. Under **Execution role**, choose **Create a new role with basic Lambda permissions**.

   In Step 5 later in this tutorial, you attach the **AmazonS3ObjectLambdaExecutionRolePolicy** to this Lambda function's execution role.
7. Keep the remaining settings set to the defaults.
8. Choose **Create function**.

Deploy your Lambda function code with .zip file archives and configure the Lambda function (console)

1. In the AWS Lambda console at https://console.aws.amazon.com/lambda/, choose **Functions** in the left navigation pane.
2. Choose the Lambda function that you created earlier (for example, **`tutorial-object-lambda-function`**).
3. On the Lambda function's details page, choose the **Code** tab. In the **Code Source** section, choose **Upload from** and then **.zip file**.

4. Choose **Upload** to select your local `.zip` file.
5. Choose the `lambda.zip` file that you created earlier, and then choose **Open**.
6. Choose **Save**.
7. In the **Runtime settings** section, choose **Edit**.
8. On the **Edit runtime settings** page, confirm that **Runtime** is set to **Python 3.8** or a later version.
9. To tell the Lambda runtime which handler method in your Lambda function code to invoke, enter **`transform.lambda_handler`** for **Handler**.

   When you configure a function in Python, the value of the handler setting is the file name and the name of the handler module, separated by a dot. For example, `transform.lambda_handler` calls the `lambda_handler` method defined in the `transform.py` file.
10. Choose **Save**.
11. (Optional) On your Lambda function's details page, choose the **Configuration** tab. In the left navigation pane, choose **General configuration**, then choose **Edit**. In the **Timeout** field, enter **1** min **0** sec. Keep the remaining settings set to the defaults, and choose **Save**.

    Timeout is the amount of time that Lambda allows a function to run for an invocation before stopping it. The default is 3 seconds. The maximum duration for a Lambda function used by S3 Object Lambda is 60 seconds. Pricing is based on the amount of memory configured and the amount of time that your code runs.

Step 5: Configure an IAM policy for your Lambda function's execution role

To enable your Lambda function to provide customized data and response headers to the `GetObject` caller, your Lambda function's execution role must have IAM permissions to call the `WriteGetObjectResponse` API.

To attach an IAM policy to your Lambda function role

1. In the AWS Lambda console at https://console.aws.amazon.com/lambda/, choose **Functions** in the left navigation pane.
2. Choose the function that you created in Step 4 (for example, **`tutorial-object-lambda-function`**).

3. On your Lambda function's details page, choose the **Configuration** tab, and then choose **Permissions** in the left navigation pane.
4. Under **Execution role**, choose the link of the **Role name**. The IAM console opens.
5. On the IAM console's **Summary** page for your Lambda function's execution role, choose the **Permissions** tab. Then, from the **Add Permissions** menu, choose **Attach policies**.
6. On the **Attach Permissions** page, enter **`AmazonS3ObjectLambdaExecutionRolePolicy`** in the search box to filter the list of policies. Select the check box next to the name of the **AmazonS3ObjectLambdaExecutionRolePolicy** policy.
7. Choose **Attach policies**.

Step 6: Create an S3 Object Lambda Access Point

An S3 Object Lambda Access Point provides the flexibility to invoke a Lambda function directly from an S3 GET request so that the function can process data retrieved from an S3 access point. When creating and configuring an S3 Object Lambda Access Point, you must specify the Lambda function to invoke and provide the event context in JSON format as custom parameters for Lambda to use.

To create an S3 Object Lambda Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose **Create Object Lambda Access Point**.
4. For **Object Lambda Access Point name**, enter the name that you want to use for the Object Lambda Access Point (for example, **`tutorial-object-lambda-accesspoint`**).
5. For **Supporting Access Point**, enter or browse to the standard access point that you created in Step 3 (for example, **`tutorial-access-point`**), and then choose **Choose supporting Access Point**.
6. For **S3 APIs**, to retrieve objects from the S3 bucket for Lambda function to process, select **GetObject**.
7. For **Invoke Lambda function**, you can choose either of the following two options for this tutorial.

- Choose **Choose from functions in your account**, and then choose the Lambda function that you created in Step 4 (for example, **`tutorial-object-lambda-function`**) from the **Lambda function** dropdown list.
- Choose **Enter ARN**, and then enter the Amazon Resource Name (ARN) of the Lambda function that you created in Step 4.

8. For **Lambda function version**, choose **$LATEST** (the latest version of the Lambda function that you created in Step 4).
9. (Optional) If you need your Lambda function to recognize and process GET requests with range and part number headers, select **Lambda function supports requests using range** and **Lambda function supports requests using part numbers**. Otherwise, clear these two check boxes.

   For more information about how to use range or part numbers with S3 Object Lambda, see Working with Range and partNumber headers.
10. (Optional) Under **Payload - *optional***, add JSON text to provide your Lambda function with additional information.

    A payload is optional JSON text that you can provide to your Lambda function as input for all invocations coming from a specific S3 Object Lambda Access Point. To customize the behaviors for multiple Object Lambda Access Points that invoke the same Lambda function, you can configure payloads with different parameters, thereby extending the flexibility of your Lambda function.

    For more information about payload, see Event context format and usage.
11. (Optional) For **Request metrics - *optional***, choose **Disable** or **Enable** to add Amazon S3 monitoring to your Object Lambda Access Point. Request metrics are billed at the standard Amazon CloudWatch rate. For more information, see CloudWatch pricing.
12. Under **Object Lambda Access Point policy - *optional***, keep the default setting.

    (Optional) You can set a resource policy. This resource policy grants the `GetObject` API permission to use the specified Object Lambda Access Point.
13. Keep the remaining settings set to the defaults, and choose **Create Object Lambda Access Point**.

Step 7: View the transformed data

Now, S3 Object Lambda is ready to transform your data for your use case. In this tutorial, S3 Object Lambda transforms all the text in your object to uppercase.

Substeps

- View the transformed data in your S3 Object Lambda Access Point
- Run a Python script to print the original and transformed data

View the transformed data in your S3 Object Lambda Access Point

When you request to retrieve a file through your S3 Object Lambda Access Point, you make a `GetObject` API call to S3 Object Lambda. S3 Object Lambda invokes the Lambda function to transform your data, and then returns the transformed data as the response to the standard S3 `GetObject` API call.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose the S3 Object Lambda Access Point that you created in Step 6 (for example, **`tutorial-object-lambda-accesspoint`**).
4. On the **Objects** tab of your S3 Object Lambda Access Point, select the file that has the same name (for example, `tutorial.txt`) as the one that you uploaded to the S3 bucket in Step 2.

   This file should contain all the transformed data.
5. To view the transformed data, choose **Open** or **Download**.

Run a Python script to print the original and transformed data

You can use S3 Object Lambda with your existing applications. To do so, update your application configuration to use the new S3 Object Lambda Access Point ARN that you created in Step 6 to retrieve data from S3.

The following example Python script prints both the original data from the S3 bucket and the transformed data from the S3 Object Lambda Access Point.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose the radio button to the left of the S3 Object Lambda Access Point that you created in Step 6 (for example, **`tutorial-object-lambda-accesspoint`**).
4. Choose **Copy ARN**.
5. Save the ARN for use later.
6. Write a Python script on your local machine to print both the original data (for example, `tutorial.txt`) from your S3 Bucket and the transformed data (for example, `tutorial.txt`) from your S3 Object Lambda Access Point). You can use the following example script.

```
import boto3
from botocore.config import Config

s3 = boto3.client('s3', config=Config(signature_version='s3v4'))

def getObject(bucket, key):
    objectBody = s3.get_object(Bucket = bucket, Key = key)
    print(objectBody["Body"].read().decode("utf-8"))
    print("\n")

print('Original object from the S3 bucket:')
# Replace the two input parameters of getObject() below with
# the S3 bucket name that you created in Step 1 and
# the name of the file that you uploaded to the S3 bucket in Step 2
getObject("tutorial-bucket",
          "tutorial.txt")

print('Object transformed by S3 Object Lambda:')
# Replace the two input parameters of getObject() below with
# the ARN of your S3 Object Lambda Access Point that you saved earlier and
# the name of the file with the transformed data (which in this case is
# the same as the name of the file that you uploaded to the S3 bucket
# in Step 2)
getObject("arn:aws:s3-object-lambda:us-west-2:111122223333:accesspoint/tutorial-
object-lambda-accesspoint",
          "tutorial.txt")
```

7. Save your Python script with a custom name (for example, `tutorial_print.py`) in the folder (for example, `object-lambda`) that you created in Step 4 on your local machine.

8. In your local terminal, run the following command from the root of the directory (for example, `object-lambda`) that you created in Step 4.

```
python3 tutorial_print.py
```

You should see both the original data and the transformed data (all text as uppercase) through the terminal. For example, you should see something like the following text.

```
Original object from the S3 bucket:
Amazon S3 Object Lambda Tutorial:
You can add your own code to process data retrieved from S3 before
returning it to an application.

Object transformed by S3 Object Lambda:
AMAZON S3 OBJECT LAMBDA TUTORIAL:
YOU CAN ADD YOUR OWN CODE TO PROCESS DATA RETRIEVED FROM S3 BEFORE
RETURNING IT TO AN APPLICATION.
```

Step 8: Clean up

If you transformed your data through S3 Object Lambda only as a learning exercise, delete the AWS resources that you allocated so that you no longer accrue charges.

Substeps

- Delete the Object Lambda Access Point
- Delete the S3 access point
- Delete the execution role for your Lambda function
- Delete the Lambda function
- Delete the CloudWatch log group
- Delete the original file in the S3 source bucket
- Delete the S3 source bucket
- Delete the IAM user

Delete the Object Lambda Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose the radio button to the left of the S3 Object Lambda Access Point that you created in Step 6 (for example, **`tutorial-object-lambda-accesspoint`**).
4. Choose **Delete**.
5. Confirm that you want to delete your Object Lambda Access Point by entering its name in the text field that appears, and then choose **Delete**.

Delete the S3 access point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points**.
3. Navigate to the access point that you created in Step 3 (for example, **`tutorial-access-point`**), and choose the radio button next to the name of the access point.
4. Choose **Delete**.
5. Confirm that you want to delete your access point by entering its name in the text field that appears, and then choose **Delete**.

Delete the execution role for your Lambda function

1. Sign in to the AWS Management Console and open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. In the left navigation pane, choose **Functions**.
3. Choose the function that you created in Step 4 (for example, **`tutorial-object-lambda-function`**).
4. On your Lambda function's details page, choose the **Configuration** tab, and then choose **Permissions** in the left navigation pane.
5. Under **Execution role**, choose the link of the **Role name**. The IAM console opens.

6. On the IAM console's **Summary** page of your Lambda function's execution role, choose **Delete role**.
7. In the **Delete role** dialog box, choose **Yes, delete**.

Delete the Lambda function

1. In the AWS Lambda console at https://console.aws.amazon.com/lambda/, choose **Functions** in the left navigation pane.
2. Select the check box to the left of the name of the function that you created in Step 4 (for example, **`tutorial-object-lambda-function`**).
3. Choose **Actions**, and then choose **Delete**.
4. In the **Delete function** dialog box, choose **Delete**.

Delete the CloudWatch log group

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Log groups**.
3. Find the log group whose name ends with the Lambda function that you created in Step 4 (for example, **`tutorial-object-lambda-function`**).
4. Select the check box to the left of the name of the log group.
5. Choose **Actions**, and then choose **Delete log group(s)**.
6. In the **Delete log group(s)** dialog box, choose **Delete**.

Delete the original file in the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Bucket name** list, choose the name of the bucket that you uploaded the original file to in Step 2 (for example, **`tutorial-bucket`**).
4. Select the check box to the left of the name of the object that you want to delete (for example, `tutorial.txt`).
5. Choose **Delete**.

6. On the **Delete objects** page, in the **Permanently delete objects?** section, confirm that you want to delete this object by entering **`permanently delete`** in the text box.
7. Choose **Delete objects**.

Delete the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the radio button next to the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
4. Choose **Delete**.
5. On the **Delete bucket** page, confirm that you want to delete the bucket by entering the bucket name in the text field, and then choose **Delete bucket**.

Delete the IAM user

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Users**, and then select the check box next to the user name that you want to delete.
3. Choose **Delete**.
4. In the **Delete *`user name`*?** dialog box, enter the user name in the text input field to confirm the deletion of the user. Choose **Delete**.

Next steps

After completing this tutorial, you can customize the Lambda function for your use case to modify the data returned by standard S3 GET requests.

The following is a list of common use cases for S3 Object Lambda:

- Masking sensitive data for security and compliance.

  For more information, see Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend.

- Filtering certain rows of data to deliver specific information.
- Augmenting data with information from other services or databases.
- Converting across data formats, such as converting XML to JSON for application compatibility.
- Compressing or decompressing files as they are being downloaded.
- Resizing and watermarking images.

  For more information, see Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved.
- Implementing custom authorization rules to access data.

For more information about S3 Object Lambda, see Transforming objects with S3 Object Lambda.

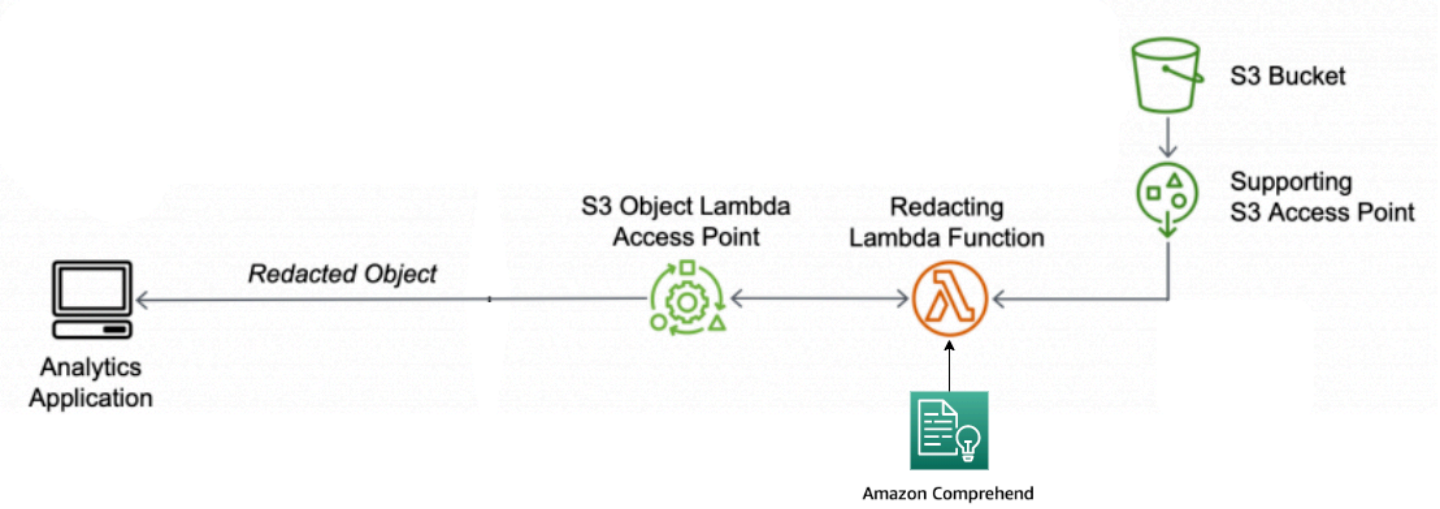
Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

When you're using Amazon S3 for shared datasets for multiple applications and users to access, it's important to restrict privileged information, such as personally identifiable information (PII), to only authorized entities. For example, when a marketing application uses some data containing PII, it might need to first mask PII data to meet data privacy requirements. Also, when an analytics application uses a production order inventory dataset, it might need to first redact customer credit card information to prevent unintended data leakage.

With S3 Object Lambda and a prebuilt AWS Lambda function powered by Amazon Comprehend, you can protect PII data retrieved from S3 before returning it to an application. Specifically, you can use the prebuilt Lambda function as a redacting function and attach it to an S3 Object Lambda Access Point. When an application (for example, an analytics application) sends standard S3 GET requests, these requests made through the S3 Object Lambda Access Point invoke the prebuilt redacting Lambda function to detect and redact PII data retrieved from an underlying data source

through a supporting S3 access point. Then, the S3 Object Lambda Access Point returns the redacted result back to the application.



In the process, the prebuilt Lambda function uses Amazon Comprehend, a natural language processing (NLP) service, to capture variations in how PII is represented, regardless of how PII exists in text (such as numerically or as a combination of words and numbers). Amazon Comprehend can even use context in the text to understand if a 4-digit number is a PIN, the last four numbers of a Social Security number (SSN), or a year. Amazon Comprehend processes any text file in UTF-8 format and can protect PII at scale without affecting accuracy. For more information, see What is Amazon Comprehend? in the *Amazon Comprehend Developer Guide*.

Objective

In this tutorial, you learn how to use S3 Object Lambda with the prebuilt Lambda function `ComprehendPiiRedactionS3ObjectLambda`. This function uses Amazon Comprehend to detect PII entities. It then redacts these entities by replacing them with asterisks. By redacting PII, you conceal sensitive data, which can help with security and compliance.

You also learn how to use and configure a prebuilt AWS Lambda function in the AWS Serverless Application Repository to work together with S3 Object Lambda for easy deployment.

Topics

- Prerequisites: Create an IAM user with permissions
- Step 1: Create an S3 bucket
- Step 2: Upload a file to the S3 bucket
- Step 3: Create an S3 access point

- Step 4: Configure and deploy a prebuilt Lambda function
- Step 5: Create an S3 Object Lambda Access Point
- Step 6: Use the S3 Object Lambda Access Point to retrieve the redacted file
- Step 7: Clean up
- Next steps

Prerequisites: Create an IAM user with permissions

Before you start this tutorial, you must have an AWS account that you can sign in to as an AWS Identity and Access Management user (IAM user) with correct permissions.

You can create an IAM user for the tutorial. To complete this tutorial, your IAM user must attach the following IAM policies to access relevant AWS resources and perform specific actions.

> **Note**
>
> For simplicity, this tutorial creates and uses an IAM user. After completing this tutorial, remember to Delete the IAM user. For production use, we recommend that you follow the Security best practices in IAM in the *IAM User Guide*. A best practice requires human users to use federation with an identity provider to access AWS with temporary credentials. Another best practice is to require workloads to use temporary credentials with IAM roles to access AWS. To learn about using AWS IAM Identity Center to create users with temporary credentials, see Getting started in the *AWS IAM Identity Center User Guide*. This tutorial also uses full-access policies. For production use, we recommend that you instead grant only the minimum permissions necessary for your use case, in accordance with security best practices.

Your IAM user requires the following AWS managed policies:

- AmazonS3FullAccess – Grants permissions to all Amazon S3 actions, including permissions to create and use an Object Lambda Access Point.
- AWSLambda_FullAccess – Grants permissions to all Lambda actions.
- AWSCloudFormationFullAccess – Grants permissions to all AWS CloudFormation actions.
- IAMFullAccess – Grants permissions to all IAM actions.

- IAMAccessAnalyzerReadOnlyAccess – Grants permissions to read all access information provided by IAM Access Analyzer.

You can directly attach these existing policies when creating an IAM user. For more information about how to create an IAM user, see Creating IAM users (console) in the *IAM User Guide*.

In addition, your IAM user requires a customer managed policy. To grant the IAM user permissions to all AWS Serverless Application Repository resources and actions, you must create an IAM policy and attach the policy to the IAM user.

To create and attach an IAM policy to your IAM user

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. On the **Visual editor** tab, for **Service**, choose **Choose a service**. Then, choose **Serverless Application Repository**.
5. For **Actions**, under **Manual actions**, select **All Serverless Application Repository actions (serverlessrepo:*)** for this tutorial.

   As a security best practice, you should allow permissions to only those actions and resources that a user needs, based on your use case. For more information, see Security best practices in IAM in the *IAM User Guide*.
6. For **Resources**, choose **All resources** for this tutorial.

   As a best practice, you should define permissions for only specific resources in specific accounts. Alternatively, you can grant least privilege using condition keys. For more information, see Grant least privilege in the *IAM User Guide*.
7. Choose **Next: Tags**.
8. Choose **Next: Review**.
9. On the **Review policy** page, enter a **Name** (for example, **`tutorial-serverless-application-repository`**) and a **Description** (optional) for the policy that you are creating. Review the policy summary to make sure that you have granted the intended permissions, and then choose **Create policy** to save your new policy.
10. In the left navigation pane, choose **Users**. Then, choose the IAM user for this tutorial.

11. On the **Summary** page of the chosen user, choose the **Permissions** tab, and then choose **Add permissions**.
12. Under **Grant permissions**, choose **Attach existing policies directly**.
13. Select the check box next to the policy that you just created (for example, **`tutorial-serverless-application-repository`**) and then choose **Next: Review**.
14. Under **Permissions summary**, review the summary to make sure that you attached the intended policy. Then, choose **Add permissions**.

Step 1: Create an S3 bucket

Create a bucket to store the original data that you plan to transform.

> **Note**
>
> Access points may be attached for another data source, such as an Amazon FSx for OpenZFS volume, however this tutorial uses a supporting access point attached to an S3 bucket.

To create a bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.

   The **Create bucket** page opens.
4. For **Bucket name**, enter a name (for example, **`tutorial-bucket`**) for your bucket.

   For more information about naming buckets in Amazon S3, see General purpose bucket naming rules.
5. For **Region**, choose the AWS Region where you want the bucket to reside.

   For more information about the bucket Region, see General purpose buckets overview.
6. For **Block Public Access settings for this bucket**, keep the default settings (**Block *all* public access** is enabled).

We recommend that you keep all Block Public Access settings enabled unless you need to turn off one or more of them for your use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

7. For the remaining settings, keep the defaults.

   (Optional) If you want to configure additional bucket settings for your specific use case, see Creating a general purpose bucket.

8. Choose **Create bucket**.

Step 2: Upload a file to the S3 bucket

Upload a text file containing known PII data of various types, such as names, banking information, phone numbers, and SSNs, to the S3 bucket as the original data that you will redact PII from later in this tutorial.

For example, you can upload following the `tutorial.txt` file. This is an example input file from Amazon Comprehend.

```
Hello Zhang Wei, I am John. Your AnyCompany Financial Services,
LLC credit card account 1111-0000-1111-0008 has a minimum payment
of $24.53 that is due by July 31st. Based on your autopay settings,
we will withdraw your payment on the due date from your
bank account number XXXXXX1111 with the routing number XXXXX0000.

Your latest statement was mailed to 100 Main Street, Any City,
WA 98121.
After your payment is received, you will receive a confirmation
text message at 206-555-0100.
If you have questions about your bill, AnyCompany Customer Service
is available by phone at 206-555-0199 or
email at support@anycompany.com.
```

To upload a file to a bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.

3. In the **Buckets** list, choose the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**) to upload your file to.
4. On the **Objects** tab for your bucket, choose **Upload**.
5. On the **Upload** page, under **Files and folders**, choose **Add files**.
6. Choose a file to upload, and then choose **Open**. For example, you can upload the `tutorial.txt` file example mentioned earlier.
7. Choose **Upload**.

Step 3: Create an S3 access point

To use an S3 Object Lambda Access Point to access and transform the original data, you must create an S3 access point and associate it with the S3 bucket that you created in Step 1. The access point must be in the same AWS Region as the objects you want to transform.

Later in this tutorial, you'll use this access point as a supporting access point for your Object Lambda Access Point.

To create an access point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points**.
3. On the **Access Points** page, choose **Create access point**.
4. In the **Access point name** field, enter the name (for example, **`tutorial-pii-access-point`**) for the access point.

   For more information about naming access points, see Naming rules for access points.
5. In the **Data source** field, enter the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**). S3 attaches the access point to this bucket.

   (Optional) You can choose **Browse S3** to browse and search the buckets in your account. If you choose **Browse S3**, choose the desired bucket, and then choose **Choose path** to populate the **Bucket name** field with that bucket's name.
6. For **Network origin**, choose **Internet**.

   For more information about network origins for access points, see Creating access points restricted to a virtual private cloud.

7. By default, all block public access settings are turned on for your access point. We recommend that you keep **Block *all* public access** enabled. For more information, see Managing public access to access points for general purpose buckets.
8. For all other access point settings, keep the default settings.

   (Optional) You can modify the access point settings to support your use case. For this tutorial, we recommend keeping the default settings.

   (Optional) If you need to manage access to your access point, you can specify an access point policy. For more information, see Policy examples for access points.
9. Choose **Create access point**.

Step 4: Configure and deploy a prebuilt Lambda function

To redact PII data, configure and deploy the prebuilt AWS Lambda function `ComprehendPiiRedactionS3ObjectLambda` for use with your S3 Object Lambda Access Point.

To configure and deploy the Lambda function

1. Sign in to the AWS Management Console and view the `ComprehendPiiRedactionS3ObjectLambda` function in the AWS Serverless Application Repository.
2. For **Application settings**, under **Application name**, keep the default value (`ComprehendPiiRedactionS3ObjectLambda`) for this tutorial.

   (Optional) You can enter the name that you want to give to this application. You might want to do this if you plan to configure multiple Lambda functions for different access needs for the same shared dataset.
3. For **MaskCharacter**, keep the default value (*****). The mask character replaces each character in the redacted PII entity.
4. For **MaskMode**, keep the default value (**MASK**). The **MaskMode** value specifies whether the PII entity is redacted with the `MASK` character or the `PII_ENTITY_TYPE` value.
5. To redact the specified types of data, for **PiiEntityTypes**, keep the default value **ALL**. The **PiiEntityTypes** value specifies the PII entity types to be considered for redaction.

   For more information about the list of supported PII entity types, see Detect Personally Identifiable Information (PII) in the *Amazon Comprehend Developer Guide*.

6. Keep the remaining settings set to the defaults.

   (Optional) If you want to configure additional settings for your specific use case, see the **Readme file** section on the left side of the page.
7. Select the check box next to **I acknowledge that this app creates custom IAM roles**.
8. Choose **Deploy**.
9. On the new application's page, under **Resources**, choose the **Logical ID** of the Lambda function that you deployed to review the function on the Lambda function page.

Step 5: Create an S3 Object Lambda Access Point

An S3 Object Lambda Access Point provides the flexibility to invoke a Lambda function directly from an S3 GET request so that the function can redact PII data retrieved from an S3 access point. When creating and configuring an S3 Object Lambda Access Point, you must specify the redacting Lambda function to invoke and provide the event context in JSON format as custom parameters for Lambda to use.

The event context provides information about the request being made in the event passed from S3 Object Lambda to Lambda. For more information about all the fields in the event context, see Event context format and usage.

To create an S3 Object Lambda Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose **Create Object Lambda Access Point**.
4. For **Object Lambda Access Point name**, enter the name that you want to use for the Object Lambda Access Point (for example, **`tutorial-pii-object-lambda-accesspoint`**).
5. For **Supporting Access Point**, enter or browse to the standard access point that you created in Step 3 (for example, **`tutorial-pii-access-point`**), and then choose **Choose supporting Access Point**.
6. For **S3 APIs**, to retrieve objects from the S3 bucket for Lambda function to process, select **GetObject**.
7. For **Invoke Lambda function**, you can choose either of the following two options for this tutorial.

- Choose **Choose from functions in your account** and choose the Lambda function that you deployed in Step 4 (for example, **`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**) from the **Lambda function** dropdown list.
- Choose **Enter ARN**, and then enter the Amazon Resource Name (ARN) of the Lambda function that you created in Step 4.

8. For **Lambda function version**, choose **$LATEST** (the latest version of the Lambda function that you deployed in Step 4).
9. (Optional) If you need your Lambda function to recognize and process GET requests with range and part number headers, select **Lambda function supports requests using range** and **Lambda function supports requests using part numbers**. Otherwise, clear these two check boxes.

   For more information about how to use range or part numbers with S3 Object Lambda, see Working with Range and partNumber headers.
10. (Optional) Under **Payload - *optional***, add JSON text to provide your Lambda function with additional information.

    A payload is optional JSON text that you can provide to your Lambda function as input for all invocations coming from a specific S3 Object Lambda Access Point. To customize the behaviors for multiple Object Lambda Access Points that invoke the same Lambda function, you can configure payloads with different parameters, thereby extending the flexibility of your Lambda function.

    For more information about payload, see Event context format and usage.
11. (Optional) For **Request metrics - *optional***, choose **Disable** or **Enable** to add Amazon S3 monitoring to your Object Lambda Access Point. Request metrics are billed at the standard Amazon CloudWatch rate. For more information, see CloudWatch pricing.
12. Under **Object Lambda Access Point policy - *optional***, keep the default setting.

    (Optional) You can set a resource policy. This resource policy grants the `GetObject` API permission to use the specified Object Lambda Access Point.
13. Keep the remaining settings set to the defaults, and choose **Create Object Lambda Access Point**.

Step 6: Use the S3 Object Lambda Access Point to retrieve the redacted file

Now, S3 Object Lambda is ready to redact PII data from your original file.

To use the S3 Object Lambda Access Point to retrieve the redacted file

When you request to retrieve a file through your S3 Object Lambda Access Point, you make a `GetObject` API call to S3 Object Lambda. S3 Object Lambda invokes the Lambda function to redact your PII data and returns the transformed data as the response to the standard S3 `GetObject` API call.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose the S3 Object Lambda Access Point that you created in Step 5 (for example, **`tutorial-pii-object-lambda-accesspoint`**).
4. On the **Objects** tab of your S3 Object Lambda Access Point, select the file that has the same name (for example, `tutorial.txt`) as the one that you uploaded to the S3 bucket in Step 2.

   This file should contain all the transformed data.
5. To view the transformed data, choose **Open** or **Download**.

   You should be able to see the redacted file, as shown in the following example.

```
Hello *********. Your AnyCompany Financial Services,
LLC credit card account ******************* has a minimum payment
of $24.53 that is due by *********. Based on your autopay settings,
we will withdraw your payment on the due date from your
bank account ********** with the routing number *********.

Your latest statement was mailed to ********************************.
After your payment is received, you will receive a confirmation
text message at ************.
If you have questions about your bill, AnyCompany Customer Service
is available by phone at ************ or
email at **********************.
```

Step 7: Clean up

If you redacted your data through S3 Object Lambda only as a learning exercise, delete the AWS resources that you allocated so that you no longer accrue charges.

Substeps

- Delete the Object Lambda Access Point
- Delete the S3 access point
- Delete the Lambda function
- Delete the CloudWatch log group
- Delete the original file in the S3 source bucket
- Delete the S3 source bucket
- Delete the IAM role for your Lambda function
- Delete the customer managed policy for your IAM user
- Delete the IAM user

Delete the Object Lambda Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Object Lambda Access Points**.
3. On the **Object Lambda Access Points** page, choose the option button to the left of the S3 Object Lambda Access Point that you created in Step 5 (for example, **`tutorial-pii-object-lambda-accesspoint`**).
4. Choose **Delete**.
5. Confirm that you want to delete your Object Lambda Access Point by entering its name in the text field that appears, and then choose **Delete**.

Delete the S3 access point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points**.

3. Navigate to the access point that you created in Step 3 (for example, **`tutorial-pii-access-point`**), and choose the option button next to the name of the access point.
4. Choose **Delete**.
5. Confirm that you want to delete your access point by entering its name in the text field that appears, and then choose **Delete**.

Delete the Lambda function

1. In the AWS Lambda console at https://console.aws.amazon.com/lambda/, choose **Functions** in the left navigation pane.
2. Choose the function that you created in Step 4 (for example, **`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**).
3. Choose **Actions**, and then choose **Delete**.
4. In the **Delete function** dialog box, choose **Delete**.

Delete the CloudWatch log group

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Log groups**.
3. Find the log group whose name ends with the Lambda function that you created in Step 4 (for example, **`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**).
4. Choose **Actions**, and then choose **Delete log group(s)**.
5. In the **Delete log group(s)** dialog box, choose **Delete**.

Delete the original file in the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Bucket name** list, choose the name of the bucket that you uploaded the original file to in Step 2 (for example, **`tutorial-bucket`**).
4. Select the check box to the left of the name of the object that you want to delete (for example, `tutorial.txt`).
5. Choose **Delete**.

6. On the **Delete objects** page, in the **Permanently delete objects?** section, confirm that you want to delete this object by entering **`permanently delete`** in the text box.
7. Choose **Delete objects**.

Delete the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the option button next to the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
4. Choose **Delete**.
5. On the **Delete bucket** page, confirm that you want to delete the bucket by entering the bucket name in the text field, and then choose **Delete bucket**.

Delete the IAM role for your Lambda function

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**, and then select the check box next to the role name that you want to delete. The role name starts with the name of the Lambda function that you deployed in Step 4 (for example, **`serverlessrepo-ComprehendPiiRedactionS3ObjectLambda`**).
3. Choose **Delete**.
4. In the **Delete** dialog box, enter the role name in the text input field to confirm deletion. Then, choose **Delete**.

Delete the customer managed policy for your IAM user

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policies**.
3. On the **Policies** page, enter the name of the customer managed policy that you created in the Prerequisites (for example, **`tutorial-serverless-application-repository`**) in the

search box to filter the list of policies. Select the option button next to the name of the policy that you want to delete.

4. Choose **Actions**, and then choose **Delete**.
5. Confirm that you want to delete this policy by entering its name in the text field that appears, and then choose **Delete**.

Delete the IAM user

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Users**, and then select the check box next to the user name that you want to delete.
3. Choose **Delete**.
4. In the **Delete *user name*?** dialog box, enter the user name in the text input field to confirm the deletion of the user. Choose **Delete**.

Next steps

After completing this tutorial, you can further explore the following related use cases:

- You can create multiple S3 Object Lambda Access Points and enable them with prebuilt Lambda functions that are configured differently to redact specific types of PII depending on the data accessors' business needs.

  Each type of user assumes an IAM role and only has access to one S3 Object Lambda Access Point (managed through IAM policies). Then, you attach each `ComprehendPiiRedactionS3ObjectLambda` Lambda function configured for a different redaction use case to a different S3 Object Lambda Access Point. For each S3 Object Lambda Access Point, you can have a supporting S3 access point to read data from an S3 bucket that stores the shared dataset.

  For more information about how to create an S3 bucket policy that allows users to read from the bucket only through S3 access points, see Configuring IAM policies for using access points.

  For more information about how to grant a user permission to access the Lambda function, the S3 access point, and the S3 Object Lambda Access Point, see Configuring IAM policies for Object Lambda Access Points.

- You can build your own Lambda function and use S3 Object Lambda with your customized Lambda function to meet your specific data needs.

  For example, to explore various data values, you can use S3 Object Lambda and your own Lambda function that uses additional Amazon Comprehend features, such as entity recognition, key phrase recognition, sentiment analysis, and document classification, to process data. You can also use S3 Object Lambda together with Amazon Comprehend Medical, a HIPAA-eligible NLP service, to analyze and extract data in a context-aware manner.

  For more information about how to transform data with S3 Object Lambda and your own Lambda function, see Tutorial: Transforming data for your application with S3 Object Lambda.

Debugging and troubleshooting S3 Object Lambda

> **Note**
>
> As of November 7th, 2025, S3 Object Lambda is available only to existing customers that are currently using the service as well as to select AWS Partner Network (APN) partners. For capabilities similar to S3 Object Lambda, learn more here - Amazon S3 Object Lambda availability change.

Requests to Amazon S3 Object Lambda access points might result in a new error response when something goes wrong with the Lambda function invocation or execution. These errors follow the same format as standard Amazon S3 errors. For information about S3 Object Lambda errors, see S3 Object Lambda Error Code List in the *Amazon Simple Storage Service API Reference*.

For more information about general Lambda function debugging, see Monitoring and troubleshooting Lambda applications in the *AWS Lambda Developer Guide*.

For information about standard Amazon S3 errors, see Error Responses in the *Amazon Simple Storage Service API Reference*.

You can enable request metrics in Amazon CloudWatch for your Object Lambda Access Points. These metrics help you monitor the operational performance of your access point. You can enable request metrics during or after creation of your Object Lambda Access Point. For more information, see S3 Object Lambda request metrics in CloudWatch.

To get more granular logging about requests made to your Object Lambda Access Points, you can enable AWS CloudTrail data events. For more information, see Logging data events for trails in the *AWS CloudTrail User Guide*.

For S3 Object Lambda tutorials, see the following:

- Tutorial: Transforming data for your application with S3 Object Lambda
- Tutorial: Detecting and redacting PII data with S3 Object Lambda and Amazon Comprehend
- Tutorial: Using S3 Object Lambda to dynamically watermark images as they are retrieved

For more information about standard access points, see Managing access to shared datasets with access points.

For information about working with buckets, see General purpose buckets overview. For information about working with objects, see Amazon S3 objects overview.

Performing object operations in bulk with Batch Operations

You can use S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. S3 Batch Operations can perform a single operation on lists of Amazon S3 objects that you specify. A single job can perform a specified operation on billions of objects containing exabytes of data. Amazon S3 tracks progress, sends notifications, and stores a detailed completion report of all actions, providing a fully managed, auditable, and serverless experience. You can use S3 Batch Operations through the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

Use S3 Batch Operations to copy objects, update server-side encryption of objects, and set object tags or access control lists (ACLs). You can also initiate object restores from S3 Glacier Flexible Retrieval or invoke an AWS Lambda function to perform custom actions using your objects. You can perform these operations on a custom list of objects, or you can use an Amazon S3 Inventory report to easily generate lists of objects. Amazon S3 Batch Operations use the same Amazon S3 API operations that you already use with Amazon S3.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets. For more

information about using Batch Operations with S3 Express One Zone and directory buckets, see Using Batch Operations with directory buckets.

S3 Batch Operations basics

You can use S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. S3 Batch Operations can run a single operation or action on lists of Amazon S3 objects that you specify.

Terminology

This section uses the terms *manifests*, *jobs*, *operations*, and *tasks*, which are defined as follows:

Manifest

A manifest is an Amazon S3 object that contains the object keys that you want Amazon S3 to act upon. If you want to create a Batch Operations job, you must supply a manifest. Your user-generated manifest must contain the bucket name, object key, and optionally, the object version for each object. If you supply a user-generated manifest, it must be in the form of an Amazon S3 Inventory report or a CSV file.

You can also have Amazon S3 generate a manifest automatically based on object filter criteria that you specify when you create your job. This option is available for Batch Operations jobs that you create by using the Amazon S3 console, or for any job type that you create by using the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

Job

A job is the basic unit of work for S3 Batch Operations. A job contains all of the information necessary to run the specified operation on the objects listed in the manifest. After you provide this information and request that the job begin, the job performs the operation for each object in the manifest.

Operation

The operation is the type of API action, such as copying objects, that you want the Batch Operations job to run. Each job performs a single type of operation across all objects that are specified in the manifest.

Task

A task is the unit of execution for a job. A task represents a single call to an Amazon S3 or AWS Lambda API operation to perform the job's operation on a single object. Over the course of a job's lifetime, S3 Batch Operations create one task for each object specified in the manifest.

How an S3 Batch Operations job works

A job is the basic unit of work for S3 Batch Operations. A job contains all of the information necessary to run the specified operation on a list of objects. To create a job, you give S3 Batch Operations a list of objects and specify the action to perform on those objects. For information about the operations that S3 Batch Operations supports, see Operations supported by S3 Batch Operations.

A batch job performs a specified operation on every object that's included in its *manifest*. A manifest lists the objects that you want a batch job to process and it is stored as an object in a bucket. You can use a comma-separated values (CSV)-formatted Cataloging and analyzing your data with S3 Inventory report as a manifest, which makes it easy to create large lists of objects located in a bucket. You can also specify a manifest in a simple CSV format that enables you to perform batch operations on a customized list of objects contained within a single bucket.

After you create a job, Amazon S3 processes the list of objects in the manifest and runs the specified operation against each object. While a job is running, you can monitor its progress programmatically or through the Amazon S3 console. You can also configure a job to generate a completion report when it finishes. The completion report describes the results of each task that was performed by the job. For more information about monitoring jobs, see Managing S3 Batch Operations jobs.

There are costs associated with S3 Batch Operations. You are billed for creating Batch Operations jobs, including jobs that are canceled before completion. For more information, see Amazon S3 pricing.

S3 Batch Operations jobs by default can process up to 4 billion objects for all operations. Specifically, Copy objects, Object Tagging, Object Lock, Invoke AWS Lambda function, Update object encryption, Replace access control list (ACL), and Batch Replication jobs can support up to 20 billion objects. There is a limit of 6 active Batch Operations jobs per AWS account. To get started creating a Batch Operations job, see Creating an S3 Batch Operations job.

S3 Batch Operations tutorial

The following tutorial presents complete end-to-end procedures for some Batch Operations tasks.

- Tutorial: Batch-transcoding videos with S3 Batch Operations

Granting permissions for Batch Operations

Before creating and running S3 Batch Operations jobs, you must grant required permissions. To create an Amazon S3 Batch Operations job, the `s3:CreateJob` user permission is required. The same entity that creates the job must also have the `iam:PassRole` permission to pass the AWS Identity and Access Management (IAM) role that's specified for the job to Batch Operations.

The following sections provide information about creating an IAM role and attaching policies. For general information about specifying IAM resources, see IAM JSON policy elements: Resource in the *IAM User Guide*.

Topics

- Creating an S3 Batch Operations IAM role
- Attaching permissions policies

Creating an S3 Batch Operations IAM role

Amazon S3 must have permissions to perform S3 Batch Operations on your behalf. You grant these permissions through an AWS Identity and Access Management (IAM) role. When you create an S3 Batch Operations job, you specify the IAM role that you want the job to use. This can be an existing IAM role. Or, if you use the Amazon S3 console to create the job, it can be an IAM role that Amazon S3 creates for you.

If you choose to let Amazon S3 create the IAM role for you, it automatically creates and attaches trust and permissions policies to the role. The trust policy allows the S3 Batch Operations service principal (`batchoperations.s3.amazonaws.com`) to assume the role. The permissions policy allows all the requisite actions for running the job, based on the settings that you specify for the job. For example, if you configure a job to copy objects from one bucket to another bucket in your AWS account, the permissions policy allows actions such as `s3:GetObject` and `s3:PutObject`. You can review the trust and permissions policies for the role before you submit the job. This option is available only if you use the Amazon S3 console to create a job, and you configure the job

to use an S3 generated object list that uses filters or is based on a replication configuration. After you submit the job, the IAM role persists in your account. You can then use it again for subsequent jobs that perform the same operation, or delete it when the job finishes running.

If you prefer to create the IAM role manually, the policy examples in this section can help you create the role. For more information about creating and configuring roles, see IAM roles in the *IAM User Guide*. For information about permissions for S3 API operations by S3 resource type, see Required permissions for Amazon S3 API operations. For additional examples, see Controlling permissions for Batch Operations using job tags and Copying objects using S3 Batch Operations.

In your IAM policies, you can also use condition keys to filter access permissions for S3 Batch Operations jobs. For more information and a complete list of Amazon S3 specific condition keys, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Trust policy

To allow the S3 Batch Operations service principal to assume the IAM role, attach the following trust policy to the role.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

Attaching permissions policies

Depending on the type of operations, you can attach one of the following policies.

Before you configure permissions, note the following:

- Regardless of the operation, Amazon S3 needs permissions to read your manifest object from your S3 bucket and optionally write a report to your bucket. Therefore, all of the following policies include these permissions.
- For Amazon S3 Inventory report manifests, S3 Batch Operations requires permission to read the manifest.json object and all associated CSV data files.
- Version-specific permissions such as `s3:GetObjectVersion` are only required when you are specifying the version ID of the objects.
- If you are running S3 Batch Operations on encrypted objects, the IAM role must also have access to the AWS KMS keys used to encrypt them.
- If you submit an inventory report manifest that's encrypted with AWS KMS, your IAM policy must include the permissions `"kms:Decrypt"` and `"kms:GenerateDataKey"` for the manifest.json object and all associated CSV data files.
- If the Batch Operations job generates a manifest in a bucket that has access control lists (ACLs) enabled and is in a different AWS account, you must grant the `s3:PutObjectAcl` permission in the IAM policy of the IAM role configured for the batch job. If you don't include this permission, the batch job fails with the error `Error occurred when preparing manifest: Failed to write manifest`.

Copy objects: PutObject

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectTagging"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
        },
        {
```

```
            "Action": [
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:ListBucket"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-source-bucket",
                "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
            ]
        }
    ]
}
```

Replace object tagging: PutObjectTagging

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
```

```
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
      ]
    }
  ]
}
```

Delete object tagging: DeleteObjectTagging

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
              "s3:DeleteObjectTagging",
              "s3:DeleteObjectVersionTagging"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
            ]
        }
    ]
}
```

Replace access control list: PutObjectAcl

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectAcl",
        "s3:PutObjectVersionAcl"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
```

```
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
      ]
    }
  ]
}
```

Restore objects: RestoreObject

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
          "s3:RestoreObject"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
```

```
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject"
      ],
      "Resource":[
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
      ]
    }
  ]
}
```

Apply Object Lock retention: PutObjectRetention

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectRetention",
                "s3:BypassGovernanceRetention"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
            ]
        }
    ]
}
```

Apply Object Lock legal hold: PutObjectLegalHold

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "s3:PutObjectLegalHold",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
            ]
        }
    ]
}
```

Replicate existing objects: InitiateReplication with an S3 generated manifest

Use this policy if you're using and storing an S3 generated manifest. For more information about using Batch Operations to replicate existing objects, see Replicating existing objects with Batch Replication.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
```

```
            "Action":[
               "s3:GetReplicationConfiguration",
               "s3:PutInventoryConfiguration"
            ],
            "Effect":"Allow",
            "Resource":[
               "arn:aws:s3:::amzn-s3-demo-source-bucket"
            ]
         },
         {
            "Action":[
               "s3:GetObject",
               "s3:GetObjectVersion"
            ],
            "Effect":"Allow",
            "Resource":[
               "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:PutObject"
            ],
            "Resource":[
               "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*",
               "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
         }
      ]
}
```

Replicate existing objects: InitiateReplication with a user manifest

Use this policy if you're using a user supplied manifest. For more information about using Batch Operations to replicate existing objects, see Replicating existing objects with Batch Replication.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
```

```
      {
         "Action":[
            "s3:InitiateReplication"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
         ]
      }
   ]
}
```

Compute checksum: Allow `GetObject`, `GetObjectVersion`, `RestoreObject`, and `PutObject`

Use this policy if you're trying to use the **Compute checksum** operation with S3 Batch Operations. Permissions for `GetObject`, `GetObjectVersion`, and `RestoreObject` are required to obtain and read the bytes of stored data. Replace the user input placeholders with your own information. For more information about **Compute checksum**, see Checking object integrity for data at rest in Amazon S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:RestoreObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket2/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket3/*"
      ]
    }
  ]
}
```

Update object encryption

You must attach the following permissions policy to allow Batch Operations to read a manifest, update the encryption type of your objects, and write a completion report. To use this permissions policy, replace the *`user input placeholders`* with your own information. For more information about using this operation and the permissions that you must attach to the role used by your IAM principal, see the section called “Update object encryption”.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [{
            "Sid": "S3BatchOperationsUpdateEncryption",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:UpdateObjectEncryption"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket-target",
                "arn:aws:s3:::amzn-s3-demo-bucket-target/*"
            ]
        },
        {
            "Sid": "S3BatchOperationsPolicyForManifestFile",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket-manifest/*"
            ]
        },
        {
            "Sid": "S3BatchOperationsPolicyForCompletionReport",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket-completion-report/*"
            ]
        },
        {
            "Sid": "S3BatchOperationsPolicyManifestGeneration",
            "Effect": "Allow",
            "Action": [
                "s3:PutInventoryConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket-target"
            ]
```

```
        },
        {
            "Sid": "AllowKMSOperationsForS3BatchOperations",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:Encrypt",
                "kms:ReEncrypt*"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:111122223333:key/01234567-89ab-
cdef-0123-456789abcdef"
            ]
        }
    ]
}
```

Creating an S3 Batch Operations job

With Amazon S3 Batch Operations, you can perform large-scale batch operations on a list of specific Amazon S3 objects. This section describes the information that you need to create an S3 Batch Operations job and the results of a `CreateJob` request. It also provides instructions for creating a Batch Operations job by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

When you create an S3 Batch Operations job, you can request a completion report for all tasks or only failed tasks. As long as at least one task has been invoked successfully, S3 Batch Operations generates a report for jobs that have been completed, have failed, or have been canceled. For more information, see Examples: S3 Batch Operations completion reports.

Topics

- Batch Operations job request elements
- Specifying a manifest
- Generating an object list automatically and saving it as a manifest file
- Creating a manifest file
- Using an existing manifest
- Creating a job
- Job responses

Batch Operations job request elements

To create an S3 Batch Operations job, you must provide the following information:

Operation

Specify the operation that you want S3 Batch Operations to run against the objects in the manifest. Each operation type accepts parameters that are specific to that operation. With Batch Operations, you can perform an operation in bulk, with the same results as if you performed that operation one-by-one on each object.

Manifest

A *manifest* is an Amazon S3 object list that contains the object keys that you want Amazon S3 to act upon. You can use the following methods to specify a manifest for a Batch Operations job:

- Direct Batch Operations to generate an object list based on metadata that you specify. You can save this list as a manifest file and use it when you create your job. This option is available for any job type that you create by using the Amazon S3 console, the AWS CLI, AWS SDKs, or Amazon S3 REST API.
- Generate an object list automatically based on an existing replication configuration. You can save this list as a manifest file and use it again for future jobs.
- Create a new manifest file manually.
- Use an existing manifest.

> **Note**
>
> - Regardless of how you specify the objects to work on, the manifest itself must be stored in a general purpose bucket. Batch Operations can't import existing manifests from, or save generated object lists as manifests to directory buckets. Objects described within the manifest, however, can be stored in directory buckets. For more information, see Directory buckets.
> - If the objects in your manifest are in a versioned bucket, specifying the version IDs for the objects directs Batch Operations to perform the operation on a specific version. If no version IDs are specified, Batch Operations performs the operation on the latest version of the objects. If your manifest includes a version ID field, you must provide a version ID for all objects in the manifest.

For more information, see Specifying a manifest.

Priority

Use job priorities to indicate the relative priority of this job to others running in your account. A higher number indicates higher priority.

Job priorities only have meaning relative to the priorities that are set for other jobs in the same account and Region. You can choose whatever numbering system works for you. For example, you might want to assign all **Restore** (`RestoreObject`) jobs a priority of 1, all **Copy** (`CopyObject`) jobs a priority of 2, and all **Replace access control lists (ACLs)** (`PutObjectAcl`) jobs a priority of 3.

S3 Batch Operations prioritizes jobs according to priority numbers, but strict ordering isn't guaranteed. Therefore, don't use job priorities to ensure that any one job starts or finishes before any other job. If you must ensure strict ordering, wait until one job has finished before starting the next.

RoleArn

Specify an AWS Identity and Access Management (IAM) role to run the job. The IAM role that you use must have sufficient permissions to perform the operation specified in the job. For example, to run a `CopyObject` job, the IAM role must have the `s3:GetObject` permission for the source bucket and the `s3:PutObject` permission for the destination bucket. The role also needs permissions to read the manifest and write the completion report.

The IAM role can be an existing role. Or, if you use the Amazon S3 console to create the job, it can be an IAM role that Amazon S3 creates automatically for you. For more information, see Granting permissions for Batch Operations.

For more information about IAM roles, see IAM roles in the *IAM User Guide*. For more information about Amazon S3 permissions, see Policy actions for Amazon S3.

> **Note**
>
> Batch Operations jobs that perform actions on directory buckets require specific permissions. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.

Report

Specify whether you want S3 Batch Operations to generate a completion report. If you request a completion report, you must also provide the parameters for the report in this element. The following information is required:

- The bucket where you want to store the report

> **Note**
>
> The report must be stored in a general purpose bucket. Batch Operations can't save reports to directory buckets. For more information, see Directory buckets.

- The format of the report
- Whether you want the report to include the details of all tasks or only failed tasks
- An optional prefix string

If the `CreateJob.Report.ExpectedBucketOwner` field is supplied, it requires that the completion report bucket owner matches. If it doesn't match, then the job fails.

> **Note**
>
> Completion reports are always encrypted with server-side encryption with Amazon S3 managed keys (SSE-S3).

Tags (optional)

You can label and control access to your S3 Batch Operations jobs by adding *tags*. You can use tags to identify who is responsible for a Batch Operations job, or to control how users interact with Batch Operations jobs. The presence of job tags can grant or limit a user's ability to cancel a job, activate a job in the confirmation state, or change a job's priority level. For example, you could grant a user permission to invoke the `CreateJob` operation, provided that the job is created with the tag `"Department=Finance"`.

You can create jobs with tags attached to them, and you can add tags to jobs after you create them.

For more information, see the section called “Using tags”.

Description (optional)

To track and monitor your job, you can also provide a description of up to 256 characters. Amazon S3 includes this description whenever it returns information about a job or displays job details on the Amazon S3 console. You can then easily sort and filter jobs according to the descriptions that you assigned. Descriptions don't need to be unique, so you can use descriptions as categories (for example, "Weekly Log Copy Jobs") to help you track groups of similar jobs.

Specifying a manifest

A *manifest* is an Amazon S3 object list that contains the object keys that you want Amazon S3 to act upon. You can use the following methods to specify a manifest for a Batch Operations job:

- Direct Batch Operations to generate an object list based on metadata that you specify. You can save this list as a manifest and use it when you create your job. This option is available for any job type that you create by using the Amazon S3 console, the AWS CLI, AWS SDKs, or Amazon S3 REST API.
- Generate an object list automatically based on an existing replication configuration. You can save this list as a manifest and use it again for future jobs.
- Create a new manifest file manually.
- Use an existing manifest.

Note

- Amazon S3 Batch Operations does not support cross-Region object list generation.
- Regardless of how you specify the objects to work on, the manifest itself must be stored in a general purpose bucket. Batch Operations can't import existing manifests from, or save generated object lists as manifests to directory buckets. Objects described within the manifest, however, can be stored in directory buckets. For more information, see Directory buckets.

Generating an object list automatically and saving it as a manifest file

You can direct Amazon S3 to generate an object list automatically based on metadata that you specify. You can save this list as a manifest and use it when you create your job. This option is available for any job type that you create by using the Amazon S3 console, the AWS CLI, AWS SDKs, or Amazon S3 REST API.

To generate an object list automatically and save it as a manifest file, you specify the following elements as part of your job creation request:

- Information about the bucket that contains your source objects, including the bucket owner and Amazon Resource Name (ARN).
- Information about the manifest output, including a flag to create a manifest file, the output bucket owner, the ARN, the prefix, the file format, and the encryption type.
- Optional criteria to filter objects by their creation date, key name, size, encryption type, KMS key ARN, Bucket Key and storage class, In the case of replication jobs, you can also use tags to filter objects.

Object filter criteria

To filter the list of objects to be included in an automatically generated object list, you can specify the following criteria. For more information, see JobManifestGeneratorFilter in the *Amazon S3 API Reference*.

CreatedAfter

If provided, the generated manifest includes only source bucket objects that were created after this time.

CreatedBefore

If provided, the generated manifest includes only source bucket objects that were created before this time.

EligibleForReplication

If provided, the generated manifest includes objects only if they are eligible for replication according to the replication configuration on the source bucket.

KeyNameConstraint

If provided, the generated manifest includes only source bucket objects whose object keys match the string constraints specified for **MatchAnySubstring**, **MatchAnyPrefix**, and **MatchAnySuffix**.

MatchAnySubstring – If provided, the generated manifest includes objects if the specified string appears anywhere within the object key string.

MatchAnyPrefix – If provided, the generated manifest includes objects if the specified string appears at the start of the object key string.

MatchAnySuffix – If provided, the generated manifest includes objects if the specified string appears at the end of the object key string.

MatchAnyObjectEncryption

If provided, the generated object list saved as a manifest file includes only source bucket objects with the indicated server-side encryption type (SSE-S3, SSE-KMS, DSSE-KMS, SSE-C, or NOT-SSE). If you select SSE- KMS or DSSE-KMS, you can optionally further filter your results by specifying a specific KMS key ARN. If you select SSE-KMS, you can also optionally further filter your results by Bucket Key enabled status.

> **Note**
>
> To improve manifest generation performance when using the `KmsKeyArn` filter, use the filter with other object metadata filters, such as `MatchAnyPrefix`, `CreatedAfter`, or `MatchAnyStorageClass`.

MatchAnyStorageClass

If provided, the generated manifest includes only source bucket objects that are stored with the specified storage class.

ObjectReplicationStatuses

If provided, the generated manifest includes only source bucket objects that have one of the specified replication statuses.

ObjectSizeGreaterThanBytes

If provided, the generated manifest includes only source bucket objects whose file size is greater than the specified number of bytes.

ObjectSizeLessThanBytes

If provided, the generated manifest includes only source bucket objects whose file size is less than the specified number of bytes.

> **Note**
>
> You can't clone most jobs that have automatically generated object lists saved as manifests. Batch replication jobs can be cloned, except when they use the `KeyNameConstraint`, `MatchAnyStorageClass`, `ObjectSizeGreaterThanBytes`, or `ObjectSizeLessThanBytes` manifest filter criteria.

The syntax for specifying manifest criteria varies depending on the method that you use to create your job. For examples, see Creating a job.

Creating a manifest file

To create a manifest file manually, you specify the manifest object key, ETag (entity tag), and optional version ID in a CSV-formatted list. The contents of the manifest must be URL-encoded.

By default, Amazon S3 automatically uses server-side encryption with Amazon S3 managed keys (SSE-S3) to encrypt a manifest that's uploaded to an Amazon S3 bucket. Manifests that use server-side encryption with customer-provided keys (SSE-C) are not supported. Manifests that use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) are supported only when you're using CSV-formatted inventory reports. Using a manually created manifest with AWS KMS is not supported.

Your manifest must contain the bucket name, object key, and optionally, the object version for each object. Any other fields in the manifest are not used by S3 Batch Operations.

> **Note**
>
> If the objects in your manifest are in a versioned bucket, specifying the version IDs for the objects directs Batch Operations to perform the operation on a specific version. If no

version IDs are specified, Batch Operations performs the operation on the latest version of the objects. If your manifest includes a version ID field, you must provide a version ID for all objects in the manifest.

The following is an example manifest in CSV format without version IDs.

```
amzn-s3-demo-bucket1,objectkey1
amzn-s3-demo-bucket1,objectkey2
amzn-s3-demo-bucket1,objectkey3
amzn-s3-demo-bucket1,photos/jpgs/objectkey4
amzn-s3-demo-bucket1,photos/jpgs/newjersey/objectkey5
amzn-s3-demo-bucket1,object%20key%20with%20spaces
```

The following is an example manifest in CSV format that includes version IDs.

```
amzn-s3-demo-bucket1,objectkey1,PZ9ibn9D5lP6p298B7S9_ceqx1n5EJ0p
amzn-s3-demo-bucket1,objectkey2,YY_ouuAJByNW1LRBfFMfxMge7XQWxMBF
amzn-s3-demo-bucket1,objectkey3,jbo9_jhdPEyB4RrmOxWS0kU0EoNrU_oI
amzn-s3-demo-bucket1,photos/jpgs/objectkey4,6EqlikJJxLTsHsnbZbSRffn24_eh5Ny4
amzn-s3-demo-bucket1,photos/jpgs/newjersey/objectkey5,imHf3FAiRsvBW_EHB8GOu.NHunHO1gVs
amzn-s3-demo-bucket1,object%20key%20with%20spaces,9HkPvDaZY5MVbMhn6TMn1YTb5ArQAo3w
```

Using an existing manifest

You can specify an existing manifest to create a Batch Operations job by using one of the following two formats:

- **Amazon S3 Inventory report** – Must be a CSV-formatted Amazon S3 Inventory report. You must specify the `manifest.json` file that is associated with the inventory report. For more information about inventory reports, see Cataloging and analyzing your data with S3 Inventory. If the inventory report includes version IDs, S3 Batch Operations operates on the specific object versions.

  Note

  - S3 Batch Operations supports CSV *inventory reports* that are encrypted with SSE-KMS.

- If you submit an inventory report manifest that's encrypted with SSE-KMS, your IAM policy must include the permissions `"kms:Decrypt"` and `"kms:GenerateDataKey"` for the `manifest.json` object and all associated CSV data files.

- **CSV file** – Each row in the file must include the bucket name, object key, and optionally, the object version. Object keys must be URL-encoded, as shown in the following examples. The manifest must either include version IDs for all objects or omit version IDs for all objects. For more information about the CSV manifest format, see JobManifestSpec in the *Amazon Simple Storage Service API Reference*.

> **Note**
>
> S3 Batch Operations doesn't support CSV *manifest files* that are encrypted with SSE-KMS.

> **Important**
>
> When you're using a manually created manifest and a versioned bucket, we recommend that you specify the version IDs for the objects. When you create a job, S3 Batch Operations parses the entire manifest before running the job. However, it doesn't take a "snapshot" of the state of the bucket.
> Because manifests can contain billions of objects, jobs might take a long time to run, which can affect which version of an object that the job acts upon. Suppose that you overwrite an object with a new version while a job is running and you didn't specify a version ID for that object. In this case, Amazon S3 performs the operation on the latest version of the object, not on the version that existed when you created the job. The only way to avoid this behavior is to specify version IDs for the objects that are listed in the manifest.

Creating a job

You can create S3 Batch Operations jobs by using the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API.

For more information about creating a job request, see Batch Operations job request elements.

Prerequisites

Before you create a Batch Operations job, confirm that you have configured the relevant permissions. For more information, see Granting permissions for Batch Operations.

Using the S3 console

To create a Batch Operations job using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Batch Operations** on the left navigation pane of the Amazon S3 console.
3. Choose **Create job**.
4. Under **Choose Region and scope**, choose and view the AWS Region where you want to create your job.

   > **Note**
   >
   > For copy operations, you must create the job in the same Region as the destination bucket. For all other operations, you must create the job in the same Region as the objects in the manifest.

5. For **Scope**, specify the list of objects that your Batch Operations job will act upon.

   Under **Object list**, you can choose to generate a manifest using an object list, generate a manifest using a replication configuration, or use an existing manifest.

   - If you choose **Generate an object list**, an object list is automatically generated based on the source location and metadata that you specify. You can save this list as a manifest and use it again for future jobs.

     > **Note**
     >
     > To generate an object list, you must have the `s3:PutInventoryConfiguration` permission. The source bucket must be a general purpose bucket.

   - If you choose **Use an existing manifest**, you can import an object list from an existing manifest. A manifest is an S3 Inventory report or CV file that lists the specific objects that you want Batch Operations to act upon.

- If you choose **Use a replication configuration**, you can generate an object list automatically based on an existing replication configuration. You can save this list as a manifest and use it again for future jobs.

  For this example, choose **Generate an object list**.

6. For **Source account**, choose the account that owns the source objects.
7. Under **Source**, enter the path to your source, for example, `s3://amzn-s3-demo-bucket`.
8. Under **Object filters**, you can use filters to filter by any portion of the object key or to filter by the end of the object key. The **Object key filters** assist in refining the list of objects to be used in the manifest. For **Object metadata filters**, choose filters to further define the scope of objects to include in the manifest.
9. Under **Choose operation**, choose the operation type that you want to perform on all objects listed in the manifest. If your manifest references objects stored in a directory bucket, only use the copy or invoke AWS Lambda function operations. All other operations aren't supported.
10. After selecting your operation type, choose **Next**.
11. Fill out the information for **Configure additional options**.

    For **Permissions**, specify the AWS Identity and Access Management (IAM) role that you want the job to use. This can be an existing role, or a role that Amazon S3 creates automatically for you. For more information, see Granting permissions for Batch Operations. Amazon S3 can create the role for you if you configured the job to use an S3 generated object list with filters or an object list based on a replication configuration.
12. When you finish configuring the additional options, choose **Next**.
13. For **Review**, verify the settings. If you need to make changes, choose **Previous**. Otherwise, you can choose **Submit job**.

Using the AWS CLI

To create your Batch Operations job with the AWS CLI, choose one of the following examples, depending on whether you're specifying an existing manifest or generating a manifest automatically.

Specify manifest

The following example shows how to use the AWS CLI to create an S3 Batch Operations `S3PutObjectTagging` job that acts on objects that are listed in an existing manifest file.

To create a Batch Operations `S3PutObjectTagging` job by specifying a manifest

1. Use the following commands to create an AWS Identity and Access Management (IAM) role, and then create an IAM policy to assign the relevant permissions. The following role and policy grant Amazon S3 permission to add object tags, which you will need when you create the job in a subsequent step.

   a. Use the following example command to create an IAM role for Batch Operations to use. To use this example command, replace *`S3BatchJobRole`* with the name that you want to give to the role.

```
aws iam create-role \
 --role-name S3BatchJobRole \
 --assume-role-policy-document '{
   "Version": "2012-10-17"       ,
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}'
```

   Record the role's Amazon Resource Name (ARN). You will need the ARN when you create a job.

   b. Use the following example command to create an IAM policy with the necessary permissions and attach it to the IAM role that you created in the previous step. For more information about the necessary permissions, see Granting permissions for Batch Operations.

> **Note**
>
> Batch Operations jobs that perform actions on directory buckets require specific permissions. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.

To use this example command, replace the *user input placeholders* as follows:

- Replace *S3BatchJobRole* with the name of your IAM role. Make sure that this name matches the name that you used earlier.
- Replace *PutObjectTaggingBatchJobPolicy* with the name that you want to give your IAM policy.
- Replace *amzn-s3-demo-destination-bucket* with the name of the bucket that contains the objects that you want to apply tags to.
- Replace *amzn-s3-demo-manifest-bucket* with the name of the bucket that contains the manifest.
- Replace *amzn-s3-demo-completion-report-bucket* with the name of the bucket where you want the completion report to be delivered to.

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name PutObjectTaggingBatchJobPolicy \
  --policy-document '{
  "Version": "2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObjectTagging",
        "s3:PutObjectVersionTagging"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketLocation"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket",
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
```

```
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:GetBucketLocation"
      ],
      "Resource":[
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
      ]
    }
  ]
}'
```

2. Use the following example command to create an `S3PutObjectTagging` job.

   The `manifest.csv` file provides a list of bucket and object key values. The job applies the specified tags to the objects that are identified in the manifest. The `ETag` is the ETag of the `manifest.csv` object, which you can get from the Amazon S3 console. This request specifies the `no-confirmation-required` parameter, so that you can run the job without having to confirm it with the `update-job-status` command. For more information, see create-job in the *AWS CLI Command Reference*.

   To use this example command, replace the *`user input placeholders`* with your own information. Replace *`IAM-role`* with the ARN of the IAM role that you created earlier.

```
aws s3control create-job \
    --region us-west-2 \
    --account-id acct-id \
    --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"keyOne",
 "Value":"ValueOne"}] }}' \
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::amzn-s3-demo-manifest-
bucket/manifest.csv","ETag":"60e460c9d1046e73f7dde5043ac3ae85"}}' \
    --report '{"Bucket":"arn:aws:s3:::amzn-s3-demo-
completion-report-bucket","Prefix":"final-reports",
 "Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
    --priority 42 \
    --role-arn IAM-role \
    --client-request-token $(uuidgen) \
    --description "job description" \
```

```
    --no-confirmation-required
```

In response, Amazon S3 returns a job ID (for example, `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`). You will need the job ID to identify, monitor, and modify the job.

Generate manifest

The following example shows how to create an S3 Batch Operations `S3DeleteObjectTagging` job that automatically generates a manifest based on your object filter criteria. This criteria includes the creation date, key name, size, storage class, and tags.

To create a Batch Operations `S3DeleteObjectTagging` job by generating a manifest

1. Use the following commands to create an AWS Identity and Access Management (IAM) role, and then create an IAM policy to assign permissions. The following role and policy grant Amazon S3 permission to delete object tags, which you will need when you create the job in a subsequent step.

   a. Use the following example command to create an IAM role for Batch Operations to use. To use this example command, replace *`S3BatchJobRole`* with the name that you want to give to the role.

```
aws iam create-role \
 --role-name S3BatchJobRole \
 --assume-role-policy-document '{
   "Version": "2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}'
```

   Record the role's Amazon Resource Name (ARN). You will need the ARN when you create a job.

b. Use the following example command to create an IAM policy with the necessary permissions and attach it to the IAM role that you created in the previous step. For more information about the necessary permissions, see Granting permissions for Batch Operations.

> **Note**
>
> Batch Operations jobs that perform actions on directory buckets require specific permissions. For more information, see AWS Identity and Access Management (IAM) for S3 Express One Zone.

To use this example command, replace the *user input placeholders* as follows:

- Replace *S3BatchJobRole* with the name of your IAM role. Make sure that this name matches the name that you used earlier.
- Replace *DeleteObjectTaggingBatchJobPolicy* with the name that you want to give your IAM policy.
- Replace *amzn-s3-demo-destination-bucket* with the name of the bucket that contains the objects that you want to apply tags to.
- Replace *amzn-s3-demo-manifest-bucket* with the name of the bucket where you want to save the manifest.
- Replace *amzn-s3-demo-completion-report-bucket* with the name of the bucket where you want the completion report to be delivered to.

```
aws iam put-role-policy \
  --role-name S3BatchJobRole \
  --policy-name DeleteObjectTaggingBatchJobPolicy \
  --policy-document '{
  "Version": "2012-10-17",
  "Statement":[
    {
      "Effect":"Allow",
      "Action":[
        "s3:DeleteObjectTagging",
        "s3:DeleteObjectVersionTagging"
      ],
```

```
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutInventoryConfiguration"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket",
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    },
    {
      "Effect":"Allow",
      "Action":[
        "s3:PutObject",
        "s3:ListBucket"
      ],
      "Resource":[
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
        "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    }
  ]
}'
```

2. Use the following example command to create the `S3DeleteObjectTagging` job.

   In this example, the values in the `--report` section specify the bucket, prefix, format, and scope of the job report that will be generated. The `--manifest-generator` section specifies information about the source bucket that contains the objects the job will act upon, information about the manifest output list that will be generated for the job, and

filter criteria to narrow the scope of objects to be included in the manifest by creation date, name constraints, size, and storage class. The command also specifies the job's priority, IAM role, and AWS Region.

For more information, see create-job in the *AWS CLI Command Reference*.

To use this example command, replace the *`user input placeholders`* with your own information. Replace *`IAM-role`* with the ARN of the IAM role that you created earlier.

```
aws s3control create-job \
    --account-id 012345678901 \
    --operation '{
        "S3DeleteObjectTagging": {}
    }' \
    --report '{
        "Bucket":"arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
        "Prefix":"reports",
        "Format":"Report_CSV_20180820",
        "Enabled":true,
        "ReportScope":"AllTasks"
    }' \
    --manifest-generator '{
        "S3JobManifestGenerator": {
          "ExpectedBucketOwner": "012345678901",
          "SourceBucket": "arn:aws:s3:::amzn-s3-demo-source-bucket",
          "EnableManifestOutput": true,
          "ManifestOutputLocation": {
            "ExpectedManifestBucketOwner": "012345678901",
            "Bucket": "arn:aws:s3:::amzn-s3-demo-manifest-bucket",
            "ManifestPrefix": "prefix",
            "ManifestFormat": "S3InventoryReport_CSV_20211130"
          },
          "Filter": {
            "CreatedAfter": "2023-09-01",
            "CreatedBefore": "2023-10-01",
            "KeyNameConstraint": {
              "MatchAnyPrefix": [
                "prefix"
              ],
              "MatchAnySuffix": [
                "suffix"
              ]
            },
```

```
            "ObjectSizeGreaterThanBytes": 100,
            "ObjectSizeLessThanBytes": 200,
            "MatchAnyStorageClass": [
              "STANDARD",
              "STANDARD_IA"
            ]
          }
        }
      }' \
    --priority 2 \
    --role-arn IAM-role \
    --region us-east-1
```

In response, Amazon S3 returns a job ID (for example, `00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c`). You will need this job ID to identify, monitor, or modify the job.

Using the AWS SDK for Java

To create your Batch Operations job with the AWS SDK for Java, you can choose between two approaches depending on whether you're specifying an existing manifest or generating a manifest automatically:

- *Specify existing manifest:* Create an S3 Batch Operations job (such as `S3PutObjectTagging`) that acts on objects listed in an existing manifest file. This approach requires you to provide the manifest location, ETag, and format specifications.
- *Generate manifest automatically:* Create an S3 Batch Operations job (such as `s3PutObjectCopy`) that automatically generates a manifest based on object filter criteria, including creation date, key name, and size constraints.

Both approaches use the S3Control client to configure job operations, manifest specifications, job reports, IAM roles, and other job parameters including priority and confirmation requirements.

For examples of how to create S3 Batch Operations jobs with the AWS SDK for Java, see Create a batch job to copy objects in the *Amazon S3 API Reference*.

Using the REST API

You can use the REST API to create a Batch Operations job. For more information, see CreateJob in the *Amazon Simple Storage Service API Reference*.

Job responses

If the `CreateJob` request succeeds, Amazon S3 returns a job ID. The job ID is a unique identifier that Amazon S3 generates automatically so that you can identify your Batch Operations job and monitor its status.

When you create a job through the AWS CLI, AWS SDKs, or REST API, you can set S3 Batch Operations to begin processing the job automatically. The job runs as soon as it's ready instead of waiting behind higher-priority jobs.

When you create a job through the Amazon S3 console, you must review the job details and confirm that you want to run the job before Batch Operations can begin to process it. If a job remains in the suspended state for over 30 days, it will fail.

Operations supported by S3 Batch Operations

You can use S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. S3 Batch Operations can perform a single operation on lists of Amazon S3 objects that you specify. A single job can perform a specified operation on billions of objects containing exabytes of data. Amazon S3 tracks progress, sends notifications, and stores a detailed completion report of all actions, providing a fully managed, auditable, and serverless experience. You can use S3 Batch Operations through the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API.

S3 Batch Operations supports the following operations:

Copy objects

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. The Batch Operations **Copy** operation copies each object that is specified in the manifest. You can copy objects to a bucket in the same AWS Region or to a bucket in a different Region. S3 Batch Operations supports most options available through Amazon S3 for copying objects. These options include setting object metadata, setting permissions, and changing an object's storage class.

You can also use the **Copy** operation to copy existing unencrypted objects and write them back to the same bucket as encrypted objects. For more information, see Encrypting objects with Amazon S3 Batch Operations.

When you copy objects, you can change the checksum algorithm used to calculate the checksum of the object. If objects don't have an additional checksum calculated, you can also add one by

specifying the checksum algorithm for Amazon S3 to use. For more information, see Checking object integrity in Amazon S3.

For more information about copying objects in Amazon S3 and the required and optional parameters, see Copying, moving, and renaming objects in this guide and CopyObject in the *Amazon Simple Storage Service API Reference*.

Restrictions and limitations

When you're using the Batch Operations **Copy** operation, the following restrictions and limitations apply:

- All source objects must be in one bucket.
- All destination objects must be in one bucket.
- You must have read permissions for the source bucket and write permissions for the destination bucket.
- Objects to be copied can be up to 5 GB in size.
- If you try to copy objects from the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive classes to the S3 Standard storage class, you must first restore these objects. For more information, see Restoring an archived object.
- You must create your Batch Operations **Copy** jobs in the destination Region, which is the Region that you intend to copy the objects to.
- All `CopyObject` options are supported except for conditional checks on entity tags (ETags) and server-side encryption with customer-provided encryption keys (SSE-C).
- If the destination bucket is unversioned, you will overwrite any objects that have the same key names.
- Objects aren't necessarily copied in the same order as they appear in the manifest. For versioned buckets, if preserving the current or noncurrent version order is important, copy all noncurrent versions first. Then, after the first job is complete, copy the current versions in a subsequent job.
- Copying objects to the Reduced Redundancy Storage (RRS) class isn't supported.
- A single Batch Operations Copy job can support a manifest with up to 20 billion objects.

Copying objects using S3 Batch Operations

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. You can use S3 Batch Operations to create a **Copy** (`CopyObject`) job to copy objects within the same account or to a different destination account.

The following examples show how to store and use a manifest that is in a different account. The first example shows how you can use Amazon S3 Inventory to deliver the inventory report to the destination account for use during job creation. The second example shows how to use a comma-separated values (CSV) manifest in the source or destination account. The third example shows how to use the **Copy** operation to enable S3 Bucket Keys for existing objects that have been encrypted by using server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS).

Copy Operation Examples

- Using an inventory report to copy objects across AWS accounts
- Using a CSV manifest to copy objects across AWS accounts
- Using Batch Operations to enable S3 Bucket Keys for SSE-KMS

Using an inventory report to copy objects across AWS accounts

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. You can use S3 Batch Operations to create a **Copy** (`CopyObject`) job to copy objects within the same account or to a different destination account.

You can use Amazon S3 Inventory to create an inventory report and use the report to create a list (manifest) of objects to copy with S3 Batch Operations. For more information about using a CSV manifest in the source or destination account, see the section called “Using a CSV manifest to copy objects across AWS accounts”.

Amazon S3 Inventory generates inventories of the objects in a bucket. The resulting list is published to an output file. The bucket that is inventoried is called the source bucket, and the bucket where the inventory report file is stored is called the destination bucket.

The Amazon S3 Inventory report can be configured to be delivered to another AWS account. Doing so allows S3 Batch Operations to read the inventory report when the job is created in the destination account.

For more information about Amazon S3 Inventory source and destination buckets, see Source and destination buckets.

The easiest way to set up an inventory is by using the Amazon S3 console, but you can also use the Amazon S3 REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs.

The following console procedure contains the high-level steps for setting up permissions for an S3 Batch Operations job. In this procedure, you copy objects from a source account to a destination account, with the inventory report stored in the destination account.

To set up Amazon S3 Inventory for source and destination buckets owned by different accounts

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Decide on (or create) a destination manifest bucket to store the inventory report in. In this procedure, the *destination account* is the account that owns both the destination manifest bucket and the bucket that the objects are copied to.
4. Configure an inventory report for a source bucket. For information about how to use the console to configure an inventory or how to encrypt an inventory list file, see Configuring Amazon S3 Inventory.

   When you configure the inventory report, you specify the destination bucket where you want the list to be stored. The inventory report for the source bucket is published to the destination bucket. In this procedure, the *source account* is the account that owns the source bucket.

   Choose **CSV** for the output format.

   When you enter information for the destination bucket, choose **Buckets in another account**. Then enter the name of the destination manifest bucket. Optionally, you can enter the account ID of the destination account.

   After the inventory configuration is saved, the console displays a message similar to the following:

   Amazon S3 could not create a bucket policy on the destination bucket. Ask the destination bucket owner to add the following bucket policy to allow Amazon S3 to place data in that bucket.

   The console then displays a bucket policy that you can use for the destination bucket.
5. Copy the destination bucket policy that appears on the console.

6. In the destination account, add the copied bucket policy to the destination manifest bucket where the inventory report is stored.
7. Create a role in the destination account that is based on the S3 Batch Operations trust policy. For more information about this trust policy, see Trust policy.

   For more information about creating a role, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

   Enter a name for the role (the following example role uses the name *BatchOperationsDestinationRoleCOPY*). Choose the **S3** service, and then choose the **S3 Batch Operations** use case, which applies the trust policy to the role.

   Then choose **Create policy** to attach the following policy to the role. To use this policy, replace the *user input placeholders* with your own information.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsDestinationObjectCOPY",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectVersionAcl",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectTagging",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging",
        "s3:GetObjectAnnotation",
        "s3:GetObjectVersionAnnotation"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-destination-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    }
```

```
  ]
}
```

The role uses the policy to grant `batchoperations.s3.amazonaws.com` permission to read the manifest in the destination bucket. It also grants permissions to GET objects, access control lists (ACLs), tags, annotations, and versions in the source object bucket, and it grants permissions to PUT objects, ACLs, tags, and versions into the destination object bucket.

8. In the source account, create a bucket policy for the source bucket that grants the role that you created in the previous step permissions to GET objects, ACLs, tags, annotations, and versions in the source bucket. This step allows S3 Batch Operations to get objects from the source bucket through the trusted role.

   The following is an example of the bucket policy for the source account. To use this policy, replace the *`user input placeholders`* with your own information.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowBatchOperationsSourceObjectCOPY",
            "Effect": "Allow",
            "Principal": {
                "AWS":
 "arn:aws:iam::111122223333:role/BatchOperationsDestinationRoleCOPY"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetObjectAcl",
                "s3:GetObjectTagging",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging",
                "s3:GetObjectAnnotation",
                "s3:GetObjectVersionAnnotation"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
        }
    ]
}
```

9. After the inventory report is available, create an S3 Batch Operations **Copy** (`CopyObject`) job in the destination account, and choose the inventory report from the destination manifest bucket. You need the ARN for the IAM role that you created in the destination account.

   For general information about creating a job, see Creating an S3 Batch Operations job.

   For information about creating a job by using the console, see Creating an S3 Batch Operations job.

Using a CSV manifest to copy objects across AWS accounts

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. You can use S3 Batch Operations to create a **Copy** (`CopyObject`) job to copy objects within the same account or to a different destination account.

You can use a CSV manifest that's stored in the source account to copy objects across AWS accounts with S3 Batch Operations. To use an S3 Inventory report as a manifest, see the section called "Using an inventory report to copy objects across AWS accounts".

For an example of the CSV format for manifest files, see the section called "Creating a manifest file".

The following procedure shows how to set up permissions when using an S3 Batch Operations job to copy objects from a source account to a destination account with a CSV manifest file that's stored in the source account.

To use a CSV manifest to copy objects across AWS accounts

1. Create an AWS Identity and Access Management (IAM) role in the destination account that's based on the S3 Batch Operations trust policy. In this procedure, the *destination account* is the account that the objects are being copied to.

   For more information about the trust policy, see Trust policy.

   For more information about creating a role, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

   If you create the role by using the console, enter a name for the role (the following example role uses the name *`BatchOperationsDestinationRoleCOPY`*). Choose the **S3** service, and then choose the **S3 Batch Operations** use case, which applies the trust policy to the role.

Then choose **Create policy** to attach the following policy to the role. To use this policy, replace the *user input placeholders* with your own information.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsDestinationObjectCOPY",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectVersionAcl",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectTagging",
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging",
        "s3:GetObjectAnnotation",
        "s3:GetObjectVersionAnnotation"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-destination-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
      ]
    }
  ]
}
```

Using the policy, the role grants `batchoperations.s3.amazonaws.com` permission to read the manifest in the source manifest bucket. It grants permissions to GET objects, access control lists (ACLs), tags, annotations, and versions in the source object bucket. It also grants permissions to PUT objects, ACLs, tags, and versions into the destination object bucket.

2. In the source account, create a bucket policy for the bucket that contains the manifest to grant the role that you created in the previous step permissions to GET objects and versions in the source manifest bucket.

This step allows S3 Batch Operations to read the manifest by using the trusted role. Apply the bucket policy to the bucket that contains the manifest.

The following is an example of the bucket policy to apply to the source manifest bucket. To use this policy, replace the *`user input placeholders`* with your own information.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceManifestRead",
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::111122223333:user/ConsoleUserCreatingJob",
          "arn:aws:iam::111122223333:role/BatchOperationsDestinationRoleCOPY"
        ]
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
    }
  ]
}
```

This policy also grants permissions to allow a console user who is creating a job in the destination account the same permissions in the source manifest bucket through the same bucket policy.

3. In the source account, create a bucket policy for the source bucket that grants the role that you created permissions to GET objects, ACLs, tags, and versions in the source object bucket. S3 Batch Operations can then get objects from the source bucket through the trusted role.

The following is an example of the bucket policy for the bucket that contains the source objects. To use this policy, replace the *user input placeholders* with your own information.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowBatchOperationsSourceObjectCOPY",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/BatchOperationsDestinationRoleCOPY"
      },
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging",
        "s3:GetObjectAnnotation",
        "s3:GetObjectVersionAnnotation"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
    }
  ]
}
```

4. Create an S3 Batch Operations job in the destination account. You need the Amazon Resource Name (ARN) for the role that you created in the destination account. For more information about creating a job, see Creating an S3 Batch Operations job.

Using Batch Operations to enable S3 Bucket Keys for SSE-KMS

S3 Bucket Keys reduce the cost of server-side encryption with AWS Key Management Service (AWS KMS) (SSE-KMS) by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys and Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects. When you perform a `CopyObject` operation by using the REST API, AWS SDKs, or AWS CLI, you can enable or disable an S3 Bucket Key at the

object level by adding the `x-amz-server-side-encryption-bucket-key-enabled` request header with a `true` or `false` value.

When you configure an S3 Bucket Key for an object by using a `CopyObject` operation, Amazon S3 updates only the settings for that object. The S3 Bucket Key settings for the destination bucket don't change. If you submit a `CopyObject` request for an AWS KMS encrypted object to a bucket that has S3 Bucket Keys enabled, your object level operation will automatically use S3 Bucket Keys unless you disable the keys in the request header. If you don't specify an S3 Bucket Key for your object, Amazon S3 applies the S3 Bucket Key settings for the destination bucket to the object.

To encrypt your existing Amazon S3 objects, you can use S3 Batch Operations. You can use the Batch Operations **Copy** operation to copy existing unencrypted objects and write them back to the same bucket as encrypted objects. For more information, see Encrypting objects with Amazon S3 Batch Operations on the AWS Storage Blog.

In the following example, you use the Batch Operations **Copy** operation to enable S3 Bucket Keys on existing objects. For more information, see the section called "Configuring an S3 Bucket Key for an object".

Topics

- Considerations for using S3 Batch Operations to encrypt objects with S3 Bucket Keys enabled
- Prerequisites
- Step 1: Get your list of objects using Amazon S3 Inventory
- Step 2: Filter your object list with S3 Select
- Step 3: Set up and run your S3 Batch Operations job

Considerations for using S3 Batch Operations to encrypt objects with S3 Bucket Keys enabled

Consider the following issues when you use S3 Batch Operations to encrypt objects with S3 Bucket Keys enabled:

- You will be charged for S3 Batch Operations jobs, objects, and requests in addition to any charges associated with the operation that S3 Batch Operations performs on your behalf, including data transfers, requests, and other charges. For more information, see Amazon S3 pricing.
- If you use a versioned bucket, each S3 Batch Operations job performed creates new encrypted versions of your objects. It also maintains the previous versions without an S3 Bucket Key

configured. To delete the old versions, set up an S3 Lifecycle expiration policy for noncurrent versions as described in Lifecycle configuration elements.

- The copy operation creates new objects with new creation dates, which can affect lifecycle actions like archiving. If you copy all objects in your bucket, all the new copies have identical or similar creation dates. To further identify these objects and create different lifecycle rules for various data subsets, consider using object tags.

Prerequisites

Before you configure your objects to use an S3 Bucket Key, review Changes to note before enabling an S3 Bucket Key.

To use this example, you must have an AWS account and at least one S3 bucket to hold your working files and encrypted results. You might also find much of the existing S3 Batch Operations documentation useful, including the following topics:

- S3 Batch Operations basics
- Creating an S3 Batch Operations job
- Operations supported by S3 Batch Operations
- Managing S3 Batch Operations jobs

Step 1: Get your list of objects using Amazon S3 Inventory

To get started, identify the S3 bucket that contains the objects to encrypt, and get a list of its contents. An Amazon S3 Inventory report is the most convenient and affordable way to do this. The report provides the list of the objects in a bucket along with their associated metadata. In this step, the source bucket is the inventoried bucket, and the destination bucket is the bucket where you store the inventory report file. For more information about Amazon S3 Inventory source and destination buckets, see Cataloging and analyzing your data with S3 Inventory.

The easiest way to set up an inventory is by using the AWS Management Console. But you can also use the REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs. Before following these steps, be sure to sign in to the console and open the Amazon S3 console at https://console.aws.amazon.com/s3/. If you encounter permission denied errors, add a bucket policy to your destination bucket. For more information, see Grant permissions for S3 Inventory and S3 analytics.

To get a list of objects using S3 Inventory

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**, and choose a bucket that contains objects to encrypt.
3. On the **Management** tab, navigate to the **Inventory configurations** section, and choose **Create inventory configuration**.
4. Give your new inventory a name, enter the name of the destination S3 bucket, and optionally create a destination prefix for Amazon S3 to assign objects in that bucket.
5. For **Output format**, choose **CSV**.
6. (Optional) In the **Additional fields – *optional*** section, choose **Encryption** and any other report fields that interest you. Set the frequency for report deliveries to **Daily** so that the first report is delivered to your bucket sooner.
7. Choose **Create** to save your configuration.

Amazon S3 can take up to 48 hours to deliver the first report, so check back when the first report arrives. After you receive your first report, proceed to the next step to filter your S3 Inventory report's contents. If you no longer want to receive inventory reports for this bucket, delete your S3 Inventory configuration. Otherwise, Amazon S3 continues to deliver reports on a daily or weekly schedule.

An inventory list isn't a single point-in-time view of all objects. Inventory lists are a rolling snapshot of bucket items, which are eventually consistent (for example, the list might not include recently added or deleted objects). Combining S3 Inventory and S3 Batch Operations works best when you work with static objects, or with an object set that you created two or more days ago. To work with more recent data, use the ListObjectsV2 (GET bucket) API operation to build your list of objects manually. If needed, repeat the process for the next few days or until your inventory report shows the desired status for all objects.

Step 2: Filter your object list with S3 Select

After you receive your S3 Inventory report, you can filter the report's contents to list only the objects that aren't encrypted with S3 Bucket Keys enabled. If you want all your bucket's objects encrypted with S3 Bucket Keys enabled, you can ignore this step. However, filtering your S3 Inventory report at this stage saves you the time and expense of re-encrypting objects that you previously encrypted with S3 Bucket Keys enabled.

Although the following steps show how to filter by using Amazon S3 Select, you can also use Amazon Athena. To decide which tool to use, look at your S3 Inventory report's `manifest.json` file. This file lists the number of data files that are associated with that report. If the number is large, use Amazon Athena because it runs across multiple S3 objects, whereas S3 Select works on one object at a time. For more information about using Amazon S3 and Athena together, see Querying Amazon S3 Inventory with Amazon Athena and "Using Athena" in the AWS Storage Blog post Encrypting objects with Amazon S3 Batch Operations.

To filter your S3 Inventory report by using S3 Select

1. Open the `manifest.json` file from your inventory report and look at the `fileSchema` section of the JSON. This informs the query that you run on the data.

   The following JSON is an example `manifest.json` file for a CSV-formatted inventory on a bucket with versioning enabled. Depending on how you configured your inventory report, your manifest might look different.

```
  {
    "sourceBucket": "batchoperationsdemo",
    "destinationBucket": "arn:aws:s3:::amzn-s3-demo-destination-bucket",
    "version": "2021-05-22",
    "creationTimestamp": "1558656000000",
    "fileFormat": "CSV",
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker,
 BucketKeyStatus",
    "files": [
      {
        "key": "demoinv/batchoperationsdemo/DemoInventory/data/009a40e4-
f053-4c16-8c75-6100f8892202.csv.gz",
        "size": 72691,
        "MD5checksum": "c24c831717a099f0ebe4a9d1c5d3935c"
      }
    ]
  }
```

   If versioning isn't activated on the bucket, or if you choose to run the report for the latest versions, the `fileSchema` is `Bucket`, `Key`, and `BucketKeyStatus`.

If versioning *is* activated, depending on how you set up the inventory report, the `fileSchema` might include the following: `Bucket`, `Key`, `VersionId`, `IsLatest`, `IsDeleteMarker`, `BucketKeyStatus`. So pay attention to columns 1, 2, 3, and 6 when you run your query.

S3 Batch Operations needs the bucket, key, and version ID as inputs to perform the job, in addition to the field to search by, which is `BucketKeyStatus`. You don't need the `VersionID` field, but it helps to specify the `VersionID` field when you operate on a versioned bucket. For more information, see Working with objects in a versioning-enabled bucket.

2. Locate the data files for the inventory report. The `manifest.json` object lists the data files under **files**.
3. After you locate and select the data file in the S3 console, choose **Actions**, and then choose **Query with S3 Select**.
4. Keep the preset **CSV**, **Comma**, and **GZIP** fields selected, and choose **Next**.
5. To review your inventory report's format before proceeding, choose **Show file preview**.
6. Enter the columns to reference in the SQL expression field, and choose **Run SQL**. The following expression returns columns 1–3 for all objects without an S3 Bucket Key configured.

```
select s._1, s._2, s._3 from s3object s where s._6 = 'DISABLED'
```

The following are example results.

```
      batchoperationsdemo,0100059%7Ethumb.jpg,lsrtIxksLu0R0ZkYPL.LhgD5caTYn6vu
      batchoperationsdemo,0100074%7Ethumb.jpg,sd2M60g6Fdazoi6D5kNARIE7KzUibmHR
      batchoperationsdemo,0100075%7Ethumb.jpg,TLYESLnl1mXD5c4BwiOIinqFrktddkoL
      batchoperationsdemo,0200147%7Ethumb.jpg,amufzfMi_fEw0Rs99rxR_HrDFlE.l3Y0
      batchoperationsdemo,0301420%7Ethumb.jpg,9qGU2SEscL.C.c_sK89trmXYIwooABSh
      batchoperationsdemo,0401524%7Ethumb.jpg,ORnEWNuB1QhHrrYAGFsZhbyvEYJ3DUor
      batchoperationsdemo,200907200065HQ
%7Ethumb.jpg,d8LgvIVjbDR5mUVwW6pu9ahTfReyn5V4
      batchoperationsdemo,200907200076HQ
%7Ethumb.jpg,XUT25d7.gK40u_GmnupdaZg3BVx2jN40
      batchoperationsdemo,201103190002HQ
%7Ethumb.jpg,z.2sVRh0myqVi0BuIrngWlsRPQdb7qOS
```

7. Download the results, save them into a CSV format, and upload them to Amazon S3 as your list of objects for the S3 Batch Operations job.
8. If you have multiple manifest files, run **Query with S3 Select** on those also. Depending on the size of the results, you could combine the lists and run a single S3 Batch Operations job or run

each list as a separate job. To decide number of jobs to run, consider the price of running each S3 Batch Operations job.

Step 3: Set up and run your S3 Batch Operations job

Now that you have your filtered CSV lists of S3 objects, you can begin the S3 Batch Operations job to encrypt the objects with S3 Bucket Keys enabled.

A *job* refers collectively to the list (manifest) of objects provided, the operation performed, and the specified parameters. The easiest way to encrypt this set of objects with S3 Bucket Keys enabled is by using the **Copy** operation and specifying the same destination prefix as the objects listed in the manifest. In an unversioned bucket, this operation overwrites the existing objects. In a bucket with versioning turned on, this operation creates a newer, encrypted version of the objects.

As part of copying the objects, specify that Amazon S3 should encrypt the objects with SSE-KMS encryption. This job copies the objects, so all of your objects will show an updated creation date upon completion, regardless of when you originally added them to Amazon S3. Also specify the other properties for your set of objects as part of the S3 Batch Operations job, including object tags and storage class.

Substeps

- Set up your IAM policy
- Set up your Batch Operations IAM role
- Enable S3 Bucket Keys for an existing bucket
- Create your Batch Operations job
- Run your Batch Operations job

Set up your IAM policy

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policy**, and then choose **Create policy**.
3. Choose the **JSON** tab. Choose **Edit policy** and add the example IAM policy that appears in the following code block.

   After copying the policy example into your IAM Console, replace the following:

a. Replace *amzn-s3-demo-source-bucket* with the name of your source bucket to copy objects from.

b. Replace *amzn-s3-demo-destination-bucket* with the name of your destination bucket to copy objects to.

c. Replace *amzn-s3-demo-manifest-bucket*/*manifest-key* with the name of your manifest object.

d. Replace *amzn-s3-demo-completion-report-bucket* with the name of the bucket where you want to save your completion reports.

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "CopyObjectsToEncrypt",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:PutObjectTagging",
        "s3:PutObjectAcl",
        "s3:PutObjectVersionTagging",
        "s3:PutObjectVersionAcl",
        "s3:GetObject",
        "s3:GetObjectAcl",
        "s3:GetObjectTagging",
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging",
        "s3:GetObjectAnnotation",
        "s3:GetObjectVersionAnnotation"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      ]
    },
    {
      "Sid": "ReadManifest",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
```

```
        "s3:GetObjectVersion"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/manifest-key"
    },
    {
      "Sid": "WriteReport",
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
    }
  ]
}
```

4. Choose **Next: Tags**.
5. Add any tags that you want (optional), and choose **Next: Review**.
6. Add a policy name, optionally add a description, and choose **Create policy**.
7. Choose **Review policy** and **Save changes**.
8. With your S3 Batch Operations policy now complete, the console returns you to the IAM **Policies** page. Filter on the policy name, choose the button to the left of the policy name, choose **Policy actions**, and choose **Attach**.

   To attach the newly created policy to an IAM role, select the appropriate users, groups, or roles in your account and choose **Attach policy**. This takes you back to the IAM console.

Set up your Batch Operations IAM role

1. On the IAM Console, in the navigation pane, choose **Roles**, and then choose **Create role**.
2. Choose **AWS service**, **S3**, and **S3 Batch Operations**. Then choose **Next: Permissions**.
3. Start entering the name of the IAM **policy** that you just created. Select the check box by the policy name when it appears, and choose **Next: Tags**.
4. (Optional) Add tags or keep the key and value fields blank for this exercise. Choose **Next: Review**.
5. Enter a role name, and accept the default description or add your own. Choose **Create role**.
6. Ensure that the user creating the job has the permissions in the following example.

Replace *account-id* with your AWS account ID and *IAM-role-name* with the name that you plan to apply to the IAM role that you will create in the Batch Operations job creation step later. For more information, see Granting permissions for Batch Operations.

```
{
"Sid": "AddIamPermissions",
"Effect": "Allow",
"Action": [
"iam:GetRole",
"iam:PassRole"
],
"Resource": "arn:aws:iam::account-id:role/IAM-role-name"
}
```

Enable S3 Bucket Keys for an existing bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the bucket that you want to turn on an S3 Bucket Key for.
3. Choose **Properties**.
4. Under **Default encryption**, choose **Edit**.
5. Under **Encryption type**, you can choose between **Amazon S3 managed keys (SSE-S3)** and **AWS Key Management Service key (SSE-KMS)**.
6. If you chose **AWS Key Management Service key (SSE-KMS)**, under **AWS KMS key**, you can specify the AWS KMS key through one of the following options.
   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**. From the list of available keys, choose a symmetric encryption KMS key in the same Region as your bucket. Both the AWS managed key (`aws/s3`) and your customer managed keys appear in the list.
   - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and then enter your KMS key ARN in the field that appears.
   - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.
7. Under **Bucket Key**, choose **Enable**, and then choose **Save changes**.

Now that an S3 Bucket Key is enabled at the bucket level, objects that are uploaded, modified, or copied into this bucket will inherit this encryption configuration by default. This includes objects that are copied by using Amazon S3 Batch Operations.

Create your Batch Operations job

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Batch Operations**, and then choose **Create Job**.
3. Choose the **Region** where you store your objects, and choose **CSV** as the manifest type.
4. Enter the path or navigate to the CSV manifest file that you created earlier from S3 Select (or Athena) results. If your manifest contains version IDs, select that box. Choose **Next**.
5. Choose the **Copy** operation, and choose the copy destination bucket. You can keep server-side encryption disabled. As long as the bucket destination has S3 Bucket Keys enabled, the copy operation applies S3 Bucket Keys at the destination bucket.
6. (Optional) Choose a storage class and the other parameters as desired. The parameters that you specify in this step apply to all operations performed on the objects that are listed in the manifest. Choose **Next**.
7. To configure server-side encryption, follow these steps:

   a. Under **Server-side encryption**, choose one of the following:

   - To keep the bucket settings for default server-side encryption of objects when storing them in Amazon S3, choose **Do not specify an encryption key**. As long as the bucket destination has S3 Bucket Keys enabled, the copy operation applies an S3 Bucket Key at the destination bucket.

     > **Note**
     >
     > If the bucket policy for the specified destination requires objects to be encrypted before storing them in Amazon S3, you must specify an encryption key. Otherwise, copying objects to the destination will fail.

   - To encrypt objects before storing them in Amazon S3, choose **Specify an encryption key**.

   b. Under **Encryption settings**, if you choose **Specify an encryption key**, you must choose either **Use destination bucket settings for default encryption** or **Override destination bucket settings for default encryption**.

c. If you choose **Override destination bucket settings for default encryption**, you must configure the following encryption settings.

   i. Under **Encryption type**, you must choose either **Amazon S3 managed keys (SSE-S3)** or **AWS Key Management Service key (SSE-KMS)**. SSE-S3 uses one of the strongest block ciphers—256-bit Advanced Encryption Standard (AES-256) to encrypt each object. SSE-KMS provides you with more control over your key. For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3) and Using server-side encryption with AWS KMS keys (SSE-KMS).

   ii. If you choose **AWS Key Management Service key (SSE-KMS)**, under **AWS KMS key**, you can specify your AWS KMS key through one of the following options.

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose a symmetric encryption KMS key in the same Region as your bucket. Both the AWS managed key (`aws/s3`) and your customer managed keys appear in the list.
   - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
   - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

   iii. Under **Bucket Key**, choose **Enable**. The copy operation applies an S3 Bucket Key at the destination bucket.

8. Give your job a description (or keep the default), set its priority level, choose a report type, and specify the **Path to completion report destination**.
9. In the **Permissions** section, be sure to choose the Batch Operations IAM role that you defined earlier. Choose **Next**.
10. Under **Review**, verify the settings. If you want to make changes, choose **Previous**. After confirming the Batch Operations settings, choose **Create job**.

    For more information, see Creating an S3 Batch Operations job.

Run your Batch Operations job

The setup wizard automatically returns you to the S3 Batch Operations section of the Amazon S3 console. Your new job transitions from the **New** state to the **Preparing** state as S3 begins the

process. During the Preparing state, S3 reads the job's manifest, checks it for errors, and calculates the number of objects.

1. Choose the refresh button in the Amazon S3 console to check progress. Depending on the size of the manifest, reading can take minutes or hours.
2. After S3 finishes reading the job's manifest, the job moves to the **Awaiting your confirmation** state. Choose the option button to the left of the Job ID, and choose **Run job**.
3. Check the settings for the job, and choose **Run job**.

   After the job begins running, you can choose the refresh button to check progress through the console dashboard view or by selecting the specific job.
4. When the job is complete, you can view the **Successful** and **Failed** object counts to confirm that everything performed as expected. If you enabled job reports, check your job report for the exact cause of any failed operations.

   You can also perform these steps by using the AWS CLI, AWS SDKs, or Amazon S3 REST API. For more information about tracking job status and completion reports, see Tracking job status and completion reports.

For examples that show the copy operation with tags using the AWS CLI and AWS SDK for Java, see Creating a Batch Operations job with job tags used for labeling.

Compute checksums

You can use S3 Batch Operations with the **Compute checksum** operation to perform checksum calculations for objects stored in Amazon S3 at rest. The **Compute checksum** operation calculates object checksums which you can use to validate data integrity without downloading or restoring objects for stored data. You can use the **Compute checksum** operation to calculate checksums for both composite and full object checksum types, for all supported checksum algorithms.

With the **Compute checksum** operation, you can process billions of objects through a single job request. This batch operation is compatible with all S3 storage classes, regardless of object size. To create a **Compute checksum** job, use the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

When you enable server access logging, you can also receive log entries about your **Compute checksum** job. The **Compute checksum** job operation emits separate server access log events after completing the checksum computations. These log entries follow the standard S3 server

access logging format and include fields such as operation type, timestamp, error codes, and the associated **Compute checksum** job ID. This logging provides an audit trail of checksum verification activities performed on your objects, helping you track and verify data integrity operations.

> **Note**
>
> The **Compute checksum** operation doesn't support server-side encryption with customer-provided encryption keys (SSE-C) encrypted objects. However, you can use the **Compute checksum** operation with objects that are encrypted by using server-side encryption with S3 managed keys (SSE-S3), server-side encryption with AWS Key Management Service (SSE-KMS). Make sure that you've granted the proper AWS KMS permissions to perform the **Compute checksum** operation.

To get started with the **Compute checksum** operation using Batch Operations, you can either:

- Manually create a new manifest file.
- Use an existing manifest.
- Direct Batch Operations to automatically generate a manifest based on object filter criteria that you specify when you create your job.

Then, submit your **Compute checksum** job request and monitor its status. After the **Compute checksum** job finishes, you automatically receive a completion report in the specified destination bucket. This completion report contains checksum information for every object in the bucket, allowing you to verify data consistency. For more information about how to use this report to examine the job, see Tracking job status and completion reports.

For more information about **Compute checksum** capabilities and how to use **Compute checksum** in the console, see Checking object integrity for data at rest in Amazon S3. For information about how to send REST requests to **Compute checksum**, see DescribeJob and CreateJob in the *Amazon S3 API Reference*.

The following sections explain how you can get started using the **Compute checksum** operation with S3 Batch Operations.

Topics

- S3 Batch Operations Compute checksum considerations
- S3 Batch Operations completion report

S3 Batch Operations Compute checksum considerations

Before using the **Compute checksum** operation, review the following list of considerations:

- If your manifest includes a version ID field, you must provide a version ID for all objects in the manifest. If the version ID isn't specified, the **Compute checksum** request performs the operation on the latest version of the object.
- To receive **Compute checksum** operation details in your server access logs, you must first enable server access logging on the source bucket and specify a destination bucket to store the logs. The destination bucket must also exist in the same AWS Region and AWS account as the source bucket. After configuring server access logging, the **Compute checksum** operation generates log records that include standard fields such as operation type, HTTP status code, S3 error codes, timestamps, and the associated **Compute checksum** job ID. The **Compute checksum** operation runs asynchronously. As a result, the log entries use a **Compute checksum** job ID, rather than a request ID, in its log entries.
- The report generation can take up to a few hours for stored objects.
- For the following S3 Glacier storage classes, the **Compute checksum** job typically takes around a week to finish:
  - S3 Glacier Flexible Retrieval
  - S3 Glacier Deep Archive
- For buckets where the completion report will be written, you must use the bucket owner condition when running the **Compute checksum** operation. If the actual bucket owner doesn't match the expected bucket owner for the submitted job request, then the job fails. For a list of S3 operations that don't support bucket owner condition, see Restrictions and limitations.

S3 Batch Operations completion report

When you create a **Compute checksum** job, you can request an S3 Batch Operations completion report. This CSV file shows the objects, success or failure codes, outputs, and descriptions. For more information about job tracking and completion reports, see Completion reports.

Delete all object tags

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. The **Delete all object tags** operation removes all Amazon S3 object tag sets currently associated with the objects that are listed in the manifest. S3 Batch Operations doesn't support deleting tags from objects while keeping other tags in place.

If the objects in your manifest are in a versioned bucket, you can remove the tag sets from a specific version of an object. To do so, you must specify a version ID for every object in the manifest. If you don't include a version ID for an object, S3 Batch Operations removes the tag set from the latest version of every object. For more information about Batch Operations manifests, see Specifying a manifest.

For more details about object tagging, see Tagging your objects in this guide, and PutObjectTagging, GetObjectTagging, and DeleteObjectTagging in the *Amazon Simple Storage Service API Reference*.

> **Warning**
>
> Running this job removes all object tag sets on every object listed in the manifest.

To use the console to create a **Delete all object tags** job, see Creating an S3 Batch Operations job.

Restrictions and limitations

When you're using Batch Operations to delete object tags, the following restrictions and limitations apply:

- The AWS Identity and Access Management (IAM) role that you specify to run the job must have permissions to perform the underlying Amazon S3 `DeleteObjectTagging` operation. For more information, see DeleteObjectTagging in the *Amazon Simple Storage Service API Reference*.
- S3 Batch Operations uses the Amazon S3 DeleteObjectTagging operation to remove the tag sets from every object in the manifest. All restrictions and limitations that apply to the underlying operation also apply to S3 Batch Operations jobs.
- A single delete object tagging job can support a manifest with up to 20 billion objects.

Invoke AWS Lambda function

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. The **Invoke AWS Lambda function** Batch Operations operation initiates AWS Lambda functions to perform custom actions on objects that are listed in a manifest. This section describes how to create a Lambda function to use with S3 Batch Operations and how to create a job to invoke the function. The S3 Batch Operations job uses the `LambdaInvoke` operation to run a Lambda function on every object listed in a manifest.

You can work with S3 Batch Operations by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API. For more information about using Lambda, see Getting Started with AWS Lambda in the *AWS Lambda Developer Guide*.

The following sections explain how you can get started using S3 Batch Operations with Lambda.

Topics

- Using Lambda with Batch Operations
- Creating a Lambda function to use with S3 Batch Operations
- Creating an S3 Batch Operations job that invokes a Lambda function
- Providing task-level information in Lambda manifests
- S3 Batch Operations tutorial

Using Lambda with Batch Operations

When using S3 Batch Operations with AWS Lambda, you must create new Lambda functions specifically for use with S3 Batch Operations. You can't reuse existing Amazon S3 event-based functions with S3 Batch Operations. Event functions can only receive messages; they don't return messages. The Lambda functions that are used with S3 Batch Operations must accept and return messages. For more information about using Lambda with Amazon S3 events, see Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

You create an S3 Batch Operations job that invokes your Lambda function. The job runs the same Lambda function on all of the objects listed in your manifest. You can control what versions of your Lambda function to use while processing the objects in your manifest. S3 Batch Operations support unqualified Amazon Resource Names (ARNs), aliases, and specific versions. For more information, see Introduction to AWS Lambda Versioning in the *AWS Lambda Developer Guide*.

If you provide the S3 Batch Operations job with a function ARN that uses an alias or the `$LATEST` qualifier, and you update the version that either of those points to, S3 Batch Operations starts calling the new version of your Lambda function. This can be useful when you want to update functionality part of the way through a large job. If you don't want S3 Batch Operations to change the version that's used, provide the specific version in the `FunctionARN` parameter when you create your job.

A single AWS Lambda job with S3 Batch Operations can support a manifest with up to 20 billion objects.

Using Lambda and Batch Operations with directory buckets

Directory buckets are a type of Amazon S3 bucket that's designed for workloads or performance-critical applications that require consistent single-digit millisecond latency. For more information, see Directory buckets.

There are special requirements for using Batch Operations to invoke Lambda functions that act on directory buckets. For example, you must structure your Lambda request using an updated JSON schema, and specify InvocationSchemaVersion 2.0 (not 1.0) when you create the job. This updated schema allows you to specify optional key-value pairs for UserArguments, which you can use to modify certain parameters of existing Lambda functions. For more information, see Automate object processing in Amazon S3 directory buckets with S3 Batch Operations and AWS Lambda in the AWS Storage Blog.

Response and result codes

S3 Batch Operations invokes the Lambda function with one or more keys, each of which has a `TaskID` associated with it. S3 Batch Operations expects a per-key result code from Lambda functions. Any task IDs sent in the request which aren't returned with a per-key result code will be given the result code from the `treatMissingKeysAs` field. `treatMissingKeysAs` is an optional request field and defaults to `TemporaryFailure`. The following table contains the other possible result codes and values for the `treatMissingKeysAs` field.

Response code	Description
`Succeeded`	The task completed normally. If you requested a job completion report, the task's result string is included in the report.
`TemporaryFailure`	The task suffered a temporary failure and will be redriven before the job completes. The result string is ignored. If this is the final redrive, the error message is included in the final report.
`PermanentFailure`	The task suffered a permanent failure. If you requested a job-completion report, the task is marked as `Failed` and includes the error

Response code	Description
	message string. Result strings from failed tasks are ignored.

Creating a Lambda function to use with S3 Batch Operations

This section provides example AWS Identity and Access Management (IAM) permissions that you must use with your Lambda function. It also contains an example Lambda function to use with S3 Batch Operations. If you have never created a Lambda function before, see Tutorial: Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

You must create Lambda functions specifically for use with S3 Batch Operations. You can't reuse existing Amazon S3 event-based Lambda functions because Lambda functions that are used for S3 Batch Operations must accept and return special data fields.

Important

AWS Lambda functions written in Java accept either RequestHandler or RequestStreamHandler handler interfaces. However, to support S3 Batch Operations request and response format, AWS Lambda requires the `RequestStreamHandler` interface for custom serialization and deserialization of a request and response.
This interface allows Lambda to pass an InputStream and OutputStream to the Java `handleRequest` method.
Be sure to use the `RequestStreamHandler` interface when using Lambda functions with S3 Batch Operations. If you use a `RequestHandler` interface, the batch job will fail with "Invalid JSON returned in Lambda payload" in the completion report.
For more information, see Handler interfaces in the *AWS Lambda User Guide*.

Example IAM permissions

The following are examples of the IAM permissions that are necessary to use a Lambda function with S3 Batch Operations.

Example— S3 Batch Operations trust policy

The following is an example of the trust policy that you can use for the Batch Operations IAM role. This IAM role is specified when you create the job and gives Batch Operations permission to assume the IAM role.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "batchoperations.s3.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

Example— Lambda IAM policy

The following is an example of an IAM policy that gives S3 Batch Operations permission to invoke the Lambda function and read the input manifest.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "BatchOperationsLambdaPolicy",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:PutObject",
                "lambda:InvokeFunction"
            ],
```

```
            "Resource": "*"
        }
    ]
}
```

Example request and response

This section provides request and response examples for the Lambda function.

Example Request

The following is a JSON example of a request for the Lambda function.

```
{
    "invocationSchemaVersion": "1.0",
    "invocationId": "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
    "job": {
        "id": "f3cc4f60-61f6-4a2b-8a21-d07600c373ce"
    },
    "tasks": [
        {
            "taskId": "dGFza2lkZ29lc2hlcmUK",
            "s3Key": "customerImage1.jpg",
            "s3VersionId": "1",
            "s3BucketArn": "arn:aws:s3:us-east-1:0123456788:amzn-s3-demo-bucket1"
        }
    ]
}
```

Example Response

The following is a JSON example of a response for the Lambda function.

```
{
  "invocationSchemaVersion": "1.0",
  "treatMissingKeysAs" : "PermanentFailure",
  "invocationId" : "YXNkbGZqYWRmaiBhc2RmdW9hZHNmZGpmaGFzbGtkaGZza2RmaAo",
  "results": [
    {
      "taskId": "dGFza2lkZ29lc2hlcmUK",
      "resultCode": "Succeeded",
      "resultString": "[\"Mary Major", \"John Stiles\"]"
    }
```

```
  ]
}
```

Example Lambda function for S3 Batch Operations

The following example Python Lambda removes a delete marker from a versioned object.

As the example shows, keys from S3 Batch Operations are URL encoded. To use Amazon S3 with other AWS services, it's important that you URL decode the key that is passed from S3 Batch Operations.

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")


def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation. When the result code is TemporaryFailure, S3 retries the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]
```

```
   try:
       obj_key = parse.unquote_plus(task["s3Key"], encoding="utf-8")
       obj_version_id = task["s3VersionId"]
       bucket_name = task["s3BucketArn"].split(":")[-1]

       logger.info(
           "Got task: remove delete marker %s from object %s.", obj_version_id,
obj_key
       )

       try:
           # If this call does not raise an error, the object version is not a delete
           # marker and should not be deleted.
           response = s3.head_object(
               Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
           )
           result_code = "PermanentFailure"
           result_string = (
               f"Object {obj_key}, ID {obj_version_id} is not " f"a delete marker."
           )

           logger.debug(response)
           logger.warning(result_string)
       except ClientError as error:
           delete_marker = error.response["ResponseMetadata"]["HTTPHeaders"].get(
               "x-amz-delete-marker", "false"
           )
           if delete_marker == "true":
               logger.info(
                   "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
               )
               try:
                   s3.delete_object(
                       Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                   )
                   result_code = "Succeeded"
                   result_string = (
                       f"Successfully removed delete marker "
                       f"{obj_version_id} from object {obj_key}."
                   )
                   logger.info(result_string)
               except ClientError as error:
```

```
                    # Mark request timeout as a temporary failure so it will be
retried.
                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from  "
                            f"object {obj_key} timed out."
                        )
                        logger.info(result_string)
                    else:
                        raise
            else:
                raise ValueError(
                    f"The x-amz-delete-marker header is either not "
                    f"present or is not 'true'."
                )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

Creating an S3 Batch Operations job that invokes a Lambda function

When creating an S3 Batch Operations job to invoke a Lambda function, you must provide the following:

- The ARN of your Lambda function (which might include the function alias or a specific version number)
- An IAM role with permission to invoke the function
- The action parameter `LambdaInvokeFunction`

For more information about creating an S3 Batch Operations job, see Creating an S3 Batch Operations job and Operations supported by S3 Batch Operations.

The following example creates an S3 Batch Operations job that invokes a Lambda function by using the AWS CLI. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3control create-job
    --account-id account-id
    --operation  '{"LambdaInvoke": { "FunctionArn": "arn:aws:lambda:region:account-
id:function:LambdaFunctionName" } }'
    --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820","Fields":
["Bucket","Key"]},"Location":{"ObjectArn":"arn:aws:s3:::amzn-s3-demo-manifest-
bucket","ETag":"ManifestETag"}}'
    --report '{"Bucket":"arn:aws:s3:::amzn-s3-demo-
bucket","Format":"Report_CSV_20180820","Enabled":true,"Prefix":"ReportPrefix","ReportScope":"Al
    --priority 2
    --role-arn arn:aws:iam::account-id:role/BatchOperationsRole
    --region region
    --description "Lambda Function"
```

Providing task-level information in Lambda manifests

When you use AWS Lambda functions with S3 Batch Operations, you might want additional data to accompany each task or key that's operated on. For example, you might want to have both a source object key and a new object key provided. Your Lambda function could then copy the source key to a new S3 bucket under a new name. By default, Batch Operations lets you specify only the destination bucket and a list of source keys in the input manifest to your job. The following examples describe how you can include additional data in your manifest so that you can run more complex Lambda functions.

To specify per-key parameters in your S3 Batch Operations manifest to use in your Lambda function's code, use the following URL-encoded JSON format. The `key` field is passed to your Lambda function as if it were an Amazon S3 object key. But it can be interpreted by the Lambda function to contain other values or multiple keys, as shown in the following examples.

> **Note**
>
> The maximum number of characters for the `key` field in the manifest is 1,024.

Example— Manifest substituting the "Amazon S3 keys" with JSON strings

The URL-encoded version must be provided to S3 Batch Operations.

```
amzn-s3-demo-bucket,{"origKey": "object1key", "newKey": "newObject1Key"}
amzn-s3-demo-bucket,{"origKey": "object2key", "newKey": "newObject2Key"}
amzn-s3-demo-bucket,{"origKey": "object3key", "newKey": "newObject3Key"}
```

Example— Manifest URL-encoded

This URL-encoded version must be provided to S3 Batch Operations. The non-URL-encoded version does not work.

```
amzn-s3-demo-bucket,%7B%22origKey%22%3A%20%22object1key%22%2C%20%22newKey%22%3A
%20%22newObject1Key%22%7D
amzn-s3-demo-bucket,%7B%22origKey%22%3A%20%22object2key%22%2C%20%22newKey%22%3A
%20%22newObject2Key%22%7D
amzn-s3-demo-bucket,%7B%22origKey%22%3A%20%22object3key%22%2C%20%22newKey%22%3A
%20%22newObject3Key%22%7D
```

Example— Lambda function with manifest format writing results to the job report

This URL-encoded manifest example contains pipe-delimited object keys for the following Lambda function to parse.

```
amzn-s3-demo-bucket,object1key%7Clower
amzn-s3-demo-bucket,object2key%7Cupper
amzn-s3-demo-bucket,object3key%7Creverse
amzn-s3-demo-bucket,object4key%7Cdelete
```

This Lambda function shows how to parse a pipe-delimited task that's encoded into the S3 Batch Operations manifest. The task indicates which revision operation is applied to the specified object.

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.resource("s3")


def lambda_handler(event, context):
    """
    Applies the specified revision to the specified object.

    :param event: The Amazon S3 batch event that contains the ID of the object to
                  revise and the revision type to apply.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of the
             operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]
    # The revision type is packed with the object key as a pipe-delimited string.
    obj_key, revision = parse.unquote_plus(task["s3Key"], encoding="utf-8").split("|")
    bucket_name = task["s3BucketArn"].split(":")[-1]

    logger.info("Got task: apply revision %s to %s.", revision, obj_key)

    try:
        stanza_obj = s3.Bucket(bucket_name).Object(obj_key)
        stanza = stanza_obj.get()["Body"].read().decode("utf-8")
```

```
        if revision == "lower":
            stanza = stanza.lower()
        elif revision == "upper":
            stanza = stanza.upper()
        elif revision == "reverse":
            stanza = stanza[::-1]
        elif revision == "delete":
            pass
        else:
            raise TypeError(f"Can't handle revision type '{revision}'.")

        if revision == "delete":
            stanza_obj.delete()
            result_string = f"Deleted stanza {stanza_obj.key}."
        else:
            stanza_obj.put(Body=bytes(stanza, "utf-8"))
            result_string = (
                f"Applied revision type '{revision}' to " f"stanza {stanza_obj.key}."
            )

        logger.info(result_string)
        result_code = "Succeeded"
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchKey":
            result_code = "Succeeded"
            result_string = (
                f"Stanza {obj_key} not found, assuming it was deleted "
                f"in an earlier revision."
            )
            logger.info(result_string)
        else:
            result_code = "PermanentFailure"
            result_string = (
                f"Got exception when applying revision type '{revision}' "
                f"to {obj_key}: {error}."
            )
            logger.exception(result_string)
    finally:
        results.append(
            {
                "taskId": task_id,
                "resultCode": result_code,
                "resultString": result_string,
            }
```

```
        )
    return {
        "invocationSchemaVersion": invocation_schema_version,
        "treatMissingKeysAs": "PermanentFailure",
        "invocationId": invocation_id,
        "results": results,
    }
```

S3 Batch Operations tutorial

The following tutorial presents complete end-to-end procedures for some Batch Operations tasks with Lambda. In this tutorial, you learn how to set up Batch Operations to invoke a Lambda function for batch-transcoding of videos stored in an S3 source bucket. The Lambda function calls AWS Elemental MediaConvert to transcode the videos.

- Tutorial: Batch-transcoding videos with S3 Batch Operations

Replace all object tags

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. The **Replace all object tags** operation replaces the object tags on every object listed in the manifest. An object tag is a key-value pair of strings that you can use to store metadata about an object.

To create a **Replace all object tags** job, you provide a set of tags that you want to apply. S3 Batch Operations applies the same set of tags to every object. The tag set that you provide replaces whatever tag sets are already associated with the objects in the manifest. S3 Batch Operations doesn't support adding tags to objects while leaving the existing tags in place.

If the objects in your manifest are in a versioned bucket, you can apply the tag set to specific versions of every object. To do so, specify a version ID for every object in the manifest. If you don't include a version ID for any objects, S3 Batch Operations applies the tag set to the latest version of every object. For more information about Batch Operations manifests, see Specifying a manifest.

For more information about object tagging, see Tagging your objects in this guide, and see PutObjectTagging, GetObjectTagging, and DeleteObjectTagging in the *Amazon Simple Storage Service API Reference*.

To use the console to create a **Replace all object tags** job, see Creating an S3 Batch Operations job.

Restrictions and limitations

When you're using Batch Operations to replace object tags, the following restrictions and limitations apply:

- The AWS Identity and Access Management (IAM) role that you specify to run the Batch Operations job must have permissions to perform the underlying `PutObjectTagging` operation. For more information about the permissions required, see PutObjectTagging in the *Amazon Simple Storage Service API Reference*.
- S3 Batch Operations uses the Amazon S3 PutObjectTagging operation to apply tags to each object in the manifest. All restrictions and limitations that apply to the underlying operation also apply to S3 Batch Operations jobs.
- A single replace all object tags job can support a manifest with up to 20 billion objects.

Replace access control list (ACL)

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. The **Replace access control list (ACL)** operation replaces the access control lists (ACLs) for every object that's listed in the manifest. By using ACLs, you can define who can access an object and what actions they can perform.

> **Note**
>
> A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

S3 Batch Operations support custom ACLs that you define and the canned ACLs that Amazon S3 provides with a predefined set of access permissions.

If the objects in your manifest are in a versioned bucket, you can apply the ACLs to specific versions of every object. To do so, specify a version ID for every object in the manifest. If you don't include a version ID for any object, S3 Batch Operations applies the ACL to the latest version of the object.

For more information about ACLs in Amazon S3, see Access control list (ACL) overview.

S3 Block Public Access

If you want to limit public access to all objects in a bucket, we recommend using Amazon S3 Block Public Access instead of using S3 Batch Operations to apply ACLs. Block Public Access can limit public access on a per-bucket or account-wide basis with a single, simple operation that takes effect quickly. This behavior makes Amazon S3 Block Public Access a better choice when your goal is to control public access to all objects in a bucket or account. Use S3 Batch Operations only when you need to apply a customized ACL to every object in the manifest. For more information about S3 Block Public Access, see Blocking public access to your Amazon S3 storage.

S3 Object Ownership

If the objects in the manifest are in a bucket that uses the **Bucket owner enforced** setting for Object Ownership, the **Replace access control list (ACL)** operation can only specify object ACLs that grant full control to the bucket owner. In this case, the **Replace access control list (ACL)** operation can't grant object ACL permissions to other AWS accounts or groups. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

Restrictions and limitations

When you're using Batch Operations to replace ACLs, the following restrictions and limitations apply:

- The AWS Identity and Access Management (IAM) role that you specify to run the **Replace access control list (ACL)** job must have permissions to perform the underlying Amazon S3 `PutObjectAcl` operation. For more information about the permissions required, see PutObjectAcl in the *Amazon Simple Storage Service API Reference*.
- S3 Batch Operations uses the Amazon S3 `PutObjectAcl` operation to apply the specified ACL to every object in the manifest. Therefore, all restrictions and limitations that apply to the underlying `PutObjectAcl` operation also apply to S3 Batch Operations **Replace access control list (ACL)** jobs.

- A single replace access control list job can support a manifest with up to 20 billion objects.

Restore objects with Batch Operations

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. The **Restore** operation initiates restore requests for the archived Amazon S3 objects that are listed in your manifest. The following archived objects must be restored before they can be accessed in real time:

- Objects archived in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes
- Objects archived through the S3 Intelligent-Tiering storage class in the Archive Access or Deep Archive Access tiers

Using a **Restore** (S3InitiateRestoreObjectOperation) operation in your S3 Batch Operations job results in a `RestoreObject` request for every object that's specified in the manifest.

> **Important**
>
> The **Restore** job only *initiates* the request to restore objects. S3 Batch Operations reports the job as complete for each object after the request is initiated for that object. Amazon S3 doesn't update the job or otherwise notify you when the objects have been restored. However, you can use S3 Event Notifications to receive notifications when the objects are available in Amazon S3. For more information, see Amazon S3 Event Notifications.

When you create a **Restore** job, the following arguments are available:

ExpirationInDays

This argument specifies how long the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive object remains available in Amazon S3. **Restore** jobs that target S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive objects require that you set `ExpirationInDays` to 1 or greater.

> **Important**
>
> Don't set `ExpirationInDays` when creating **Restore** operation jobs that target S3 Intelligent-Tiering Archive Access and Deep Archive Access tier objects. Objects in S3

Intelligent-Tiering archive access tiers aren't subject to restore expiration, so specifying `ExpirationInDays` results in a `RestoreObject` request failure.

GlacierJobTier

Amazon S3 can restore objects by using one of three different retrieval tiers: `EXPEDITED`, `STANDARD`, and `BULK`. However, the S3 Batch Operations feature supports only the `STANDARD` and `BULK` retrieval tiers. For more information about the differences between the retrieval tiers, see Understanding archive retrieval options.

For more information about the pricing for each tier, see the **Requests & data retrievals** section on the Amazon S3 pricing page.

Differences when restoring from S3 Glacier and S3 Intelligent-Tiering

Restoring archived files from the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes differs from restoring files from the S3 Intelligent-Tiering storage class in the Archive Access or Deep Archive Access tiers.

- When you restore from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, a temporary *copy* of the object is created. Amazon S3 deletes this copy after the value that you specified in the `ExpirationInDays` argument has elapsed. After this temporary copy is deleted, you must submit an additional restore request to access the object.
- When restoring archived S3 Intelligent-Tiering objects, do *not* specify the `ExpirationInDays` argument. When you restore an object from the S3 Intelligent-Tiering Archive Access or Deep Archive Access tiers, the object transitions back into the S3 Intelligent-Tiering Frequent Access tier. After a minimum of 90 consecutive days of no access, the object automatically transitions into the Archive Access tier. After a minimum of 180 consecutive days of no access, the object automatically moves into the Deep Archive Access tier.
- Batch Operations jobs can operate either on S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage class objects *or* on S3 Intelligent-Tiering Archive Access and Deep Archive Access storage tier objects. Batch Operations can't operate on both types of archived objects in the same job. To restore objects of both types, you *must* create separate Batch Operations jobs.

Overlapping restores

If your S3InitiateRestoreObjectOperation job tries to restore an object that's already in the process of being restored, S3 Batch Operations proceeds as follows.

The restore operation succeeds for the object if either of the following conditions is true:

- Compared to the restoration request already in progress, this job's `ExpirationInDays` value is the same and its `GlacierJobTier` value is faster.
- The previous restoration request has already been completed, and the object is currently available. In this case, Batch Operations updates the expiration date of the restored object to match the `ExpirationInDays` value that's specified in the in-progress restoration request.

The restore operation fails for the object if any of the following conditions are true:

- The restoration request already in progress hasn't yet been completed, and the restoration duration for this job (specified by the `ExpirationInDays` value) is different from the restoration duration that's specified in the in-progress restoration request.
- The restoration tier for this job (specified by the `GlacierJobTier` value) is the same or slower than the restoration tier that's specified in the in-progress restoration request.

Limitations

`S3InitiateRestoreObjectOperation` jobs have the following limitations:

- You must create the job in the same Region as the archived objects.
- S3 Batch Operations doesn't support the `EXPEDITED` retrieval tier.
- A single Batch Operations Restore job can support a manifest with up to 4 billion objects.

For more information about restoring objects, see Restoring an archived object.

Update object encryption

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. The Batch Operations `UpdateObjectEncryption` operation updates the server-side encryption type of more than one Amazon S3 object with a single request. A single `UpdateObjectEncryption` operation job can support a manifest with up to 20 billion objects.

The `UpdateObjectEncryption` operation is supported for all Amazon S3 storage classes that are supported by general purpose buckets. You can use the `UpdateObjectEncryption` operation to change encrypted objects from server-side encryption with Amazon S3 managed keys (SSE- S3) to AWS Key Management Service (AWS KMS) keys (SSE-KMS), or to apply S3 Bucket Keys. You can also use the `UpdateObjectEncryption` operation to change the customer-managed KMS key used to encrypt your data so that you can comply with custom key-rotation standards.

When you create a Batch Operations job, you can generate an object list based on the source location and filter criteria that you specify. You can use the `MatchAnyObjectEncryption` filter to generate a list of objects from your bucket that you want to update and include in your manifest. The generated object list includes only source bucket objects with the indicated server-side encryption type. If you select SSE-KMS, you can optionally further filter your results by specifying a specific KMS Key ARN and Bucket Key enabled status. For more information, see `JobManifestGeneratorFilter` and `SSEKMSFilter` in the *Amazon S3API Reference*.

Restrictions and considerations

When you're using the Batch Operations `UpdateObjectEncryption` operation, the following restrictions and considerations apply:

- The `UpdateObjectEncryption` operation doesn't support objects that are unencrypted or objects that are encrypted with either dual-layer server-side encryption with AWS KMS keys (DSSE-KMS) or customer-provided encryption keys (SSE-C). Additionally, you cannot specify SSE-S3 encryption type `UpdateObjectEncryption` request.
- You can use the `UpdateObjectEncryption` operation to update objects in buckets that have S3 Versioning enabled. To update the encryption type of a particular version, you must specify a version ID in your `UpdateObjectEncryption` request. If you don't specify version ID, the `UpdateObjectEncryption` request acts on the current version of the object. For more information about S3 Versioning, see the section called “Retaining multiple versions of objects”.
- The `UpdateObjectEncryption` operation fails on any object that has an S3 Object Lock retention mode or legal hold applied to it. If an object has a governance-mode retention period or a legal hold, you must first remove the Object Lock status on the object before you issue your `UpdateObjectEncryption` request. You can't use the `UpdateObjectEncryption` operation with objects that have an Object Lock compliance mode retention period applied to them. For more information about S3 Object Lock, see the section called “Locking objects”.
- `UpdateObjectEncryption` requests on source buckets with live replication enabled won't initiate replica events in the destination bucket. If you want to change the encryption

type of objects in both your source and destination buckets, you must initiate separate `UpdateObjectEncryption` requests on the objects in the source and destination buckets.

- By default, all `UpdateObjectEncryption` requests that specify a customer-managed KMS key are restricted to KMS keys that are owned by the bucket owner's AWS account. If you're using AWS Organizations, you can request the ability to use AWS KMS keys owned by other member accounts within your organization by contacting AWS Support.
- If you use S3 Batch Replication to replicate datasets cross region and your objects previously had their server-side encryption type updated from SSE-S3 to SSE-KMS, you may need additional permissions. On the source region bucket, you must have `kms:decrypt` permissions. Then, you will need the `kms:decrypt` and `kms:encrypt` permissions for the bucket in the destination region.
- Provide a full KMS key ARN in your `UpdateObjectEncryption` request. You can't use an alias name or alias ARN. You can determine the full KMS Key ARN in the AWS KMS Console or using the AWS KMS `DescribeKey` API.
- To improve manifest generation performance when using the `KmsKeyArn` filter, use this filter in conjunction with other object metadata filters. For example, you can combine `KmsKeyArn` with `MatchAnyPrefix`, `CreatedAfter`, or `MatchAnyStorageClass` when you automatically generate a manifest in S3 Batch Operations.

For more information about `UpdateObjectEncryption`, see the section called “Updating encryption”.

Required permissions

To perform the `UpdateObjectEncryption` operation, add the following AWS Identity and Access Management (IAM) policy to your IAM principal (user, role, or group). To use this policy, replace *`amzn-s3-demo-bucket`* with the name of the bucket that contains the objects that you want to update encryption for. Replace *`amzn-s3-demo-manifest-bucket`* with the name of the bucket that contains your manifest, and replace *`amzn-s3-demo-completion-report-bucket`* with the name of the bucket where you want to store your completion report.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3BatchOperationsUpdateEncryption",
            "Effect": "Allow",
```

```
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:UpdateObjectEncryption"
            ],
            "Resource": [
                 "arn:aws:s3:::amzn-s3-demo-bucket-target",
                "arn:aws:s3:::amzn-s3-demo-bucket-target-target/*"
            ]
        },
        {
            "Sid": "S3BatchOperationsPolicyForManifestFile",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket-manifest/*"
            ]
        },
        {
            "Sid": "S3BatchOperationsPolicyForCompletionReport",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket-completion-report/*"
            ]
        },
        {
            "Sid": "S3BatchOperationsPolicyManifestGeneration",
            "Effect": "Allow",
            "Action": [
                "s3:PutInventoryConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket-target"
            ]
        },
        {
            "Sid": "AllowKMSOperationsForS3BatchOperations",
```

```
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:Encrypt",
                "kms:ReEncrypt*"
            ],
            "Resource": [                "arn:aws:kms:us-
east-1:111122223333:key/01234567-89ab-cdef-0123-456789abcdef"
            ]
        }
    ]
}
```

For the trust policy and permissions policy that you must attach to the IAM role that the S3 Batch Operations service principal assumes to run Batch Operations jobs on your behalf, see the section called "Granting permissions" and the section called "Update object encryption".

Creating a Batch Operations job to update object encryption

To update the server-side encryption type of more than one Amazon S3 object with a single request, you can use S3 Batch Operations. You can use S3 Batch Operations through the Amazon S3 console, AWS Command Line Interface (AWS CLI) AWS SDKs, or the Amazon S3 REST API.

Using the AWS CLI

To run the following commands, you must have the AWS CLI installed and configured. If you don't have the AWS CLI installed, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

Alternatively, you can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? and Getting started with AWS CloudShell in the *AWS CloudShell User Guide*.

Example 1 – Create a Batch Operations job that updates encrypted objects from one AWS KMS key to another KMS key

The following example shows how to create an S3 Batch Operations job that updates the encryption settings for multiple objects in your general purpose bucket. This command creates a job that changes objects encrypted with one AWS Key Management Service (AWS KMS) key to

use a different KMS key. This job also generates and saves a manifest of the affected objects and creates a report of the results. To use this command, replace the *user input placeholders* with your own information.

```
aws s3control create-job --account-id account-id \
--no-confirmation-required \
--operation '{"S3UpdateObjectEncryption": {  "ObjectEncryption": { "SSEKMS":
 { "KMSKeyArn": "KMS-key-ARN-to-apply", "BucketKeyEnabled": false  }  }  } }' \
--report '{ "Enabled": true, "Bucket": "report-bucket-ARN",  "Format":
 "Report_CSV_20180820", "Prefix": "report", "ReportScope": "AllTasks" }' \
--manifest-generator '{ "S3JobManifestGenerator": { "ExpectedBucketOwner":
 "account-id", "SourceBucket": "source-bucket-ARN", "EnableManifestOutput": true,
 "ManifestOutputLocation": { "Bucket": "manifest-bucket-ARN", "ManifestFormat":
 "S3InventoryReport_CSV_20211130", "ManifestPrefix": "manifest-prefix" }, "Filter":
 {   "MatchAnyObjectEncryption": [{ "SSEKMS": { "KmsKeyArn": "kms-key-ARN-to-
match" } }] } } }' \
--priority 1 \
--role-arn batch-operations-role-ARN
```

For best performance, we recommend using the `KmsKeyArn` filter in conjunction with other object metadata filters, such as `MatchAnyPrefix`, `CreatedAfter`, or `MatchAnyStorageClass`.

Example 2 – Create a Batch Operations job that updates SSE-S3 encrypted objects to SSE-KMS

The following example shows how to create an S3 Batch Operations job that updates the encryption settings for multiple objects in your general purpose bucket. This command creates a job that changes objects encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3) to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) instead. This job also generates and saves a manifest of the affected objects and creates a report of the results. To use this command, replace the *user input placeholders* with your own information.

```
aws s3control create-job --account-id account-id \
--no-confirmation-required \
--operation '{"S3UpdateObjectEncryption": {  "ObjectEncryption": { "SSEKMS":
 { "KMSKeyArn": "KMS-key-ARN-to-apply", "BucketKeyEnabled": false  }  }  } }' \
--report '{ "Enabled": true, "Bucket": "report-bucket-ARN",  "Format":
 "Report_CSV_20180820", "Prefix": "report", "ReportScope": "AllTasks" }' \
--manifest-generator '{ "S3JobManifestGenerator": { "ExpectedBucketOwner":
 "account-id", "SourceBucket": "source-bucket-ARN", "EnableManifestOutput": true,
 "ManifestOutputLocation": { "Bucket": "manifest-bucket-ARN", "ManifestFormat":
```

```
 "S3InventoryReport_CSV_20211130", "ManifestPrefix": "manifest-prefix" }, "Filter": {
  "MatchAnyObjectEncryption": [{ "SSES3": {} }] } } }' \
--priority 1 \
--role-arn batch-operations-role-ARN
```

For best performance, we recommend using the `KmsKeyArn` filter in conjunction with other object metadata filters, such as `MatchAnyPrefix`, `CreatedAfter`, or `MatchAnyStorageClass`.

S3 Object Lock retention

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. You can use the **Object Lock retention** operation to apply fixed or variable retention to your objects in either *governance* mode or *compliance* mode. These retention modes apply different levels of protection. You can apply either retention mode to any object version. Retention dates, like legal holds, prevent an object from being overwritten or deleted. Amazon S3 stores the *retain until date* specified in the object's metadata and protects the specified version of the object version until the retention period expires.

You can use S3 Batch Operations with Object Lock to manage the retention periods of many Amazon S3 objects at once. You can set a fixed retain-until-date, or set event holds and durations on existing objects at scale, or release event holds across objects. An event hold keeps an object write-once-read-many (WORM) protected until you release it. When you release the hold, Amazon S3 sets the final retain-until-date to the release time plus the configured duration. You specify the list of target objects in your manifest and submit the manifest to Batch Operations for completion. For more information, see S3 Object Lock the section called “Retention periods”.

Your S3 Batch Operations job with retention dates runs until completion, until cancellation, or until a failure state is reached. We recommend using S3 Batch Operations and S3 Object Lock retention when you want to add, change, or remove the retention date for many objects with a single request.

Batch Operations verifies that Object Lock is enabled on your bucket before processing any keys in the manifest. To perform the operations and validation, Batch Operations needs the `s3:GetBucketObjectLockConfiguration` and `s3:PutObjectRetention` permissions in an AWS Identity and Access Management (IAM) role to allow Batch Operations to call Object Lock on your behalf. For more information, see the section called “Object Lock considerations”.

For information about using this operation with the REST API, see `S3PutObjectRetention` in the CreateJob operation in the *Amazon Simple Storage Service API Reference*.

For an AWS Command Line Interface (AWS CLI) example of using this operation, see the section called "Use Batch Operations with Object Lock retention". For an AWS SDK for Java example, see the section called "Use Batch Operations with Object Lock retention".

Restrictions and limitations

When you're using Batch Operations to apply Object Lock retention periods, the following restrictions and limitations apply:

- S3 Batch Operations doesn't make any bucket-level changes.
- Versioning and S3 Object Lock must be configured on the bucket where the job is performed.
- All objects listed in the manifest must be in the same bucket.
- The operation works on the latest version of the object unless a version is explicitly specified in the manifest.
- You need `s3:PutObjectRetention` permission in your IAM role to use an **Object Lock retention** job.
- The `s3:GetBucketObjectLockConfiguration` IAM permission is required to confirm that Object Lock is enabled for the S3 bucket that you're performing the job on.
- You can only extend the retention period of objects with `COMPLIANCE` mode retention dates applied, and this retention period can't be shortened.
- A single S3 Object Lock retention job can support a manifest with up to 20 billion objects.
- When you release an event hold, you cannot specify a new retain-until-date. If you want to turn off the event hold and apply a later, explicit retain-until-date in the same job, use fixed retention instead. Fixed retention turns off the event hold and applies the retain-until-date that you specify.

S3 Object Lock legal hold

You can use Amazon S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. You can use the **Object Lock legal hold** operation to place a legal hold on an object version. Like setting a retention period, a legal hold prevents an object version from being overwritten or deleted. However, a legal hold doesn't have an associated retention period and remains in effect until it's removed.

You can use S3 Batch Operations with Object Lock to add legal holds to many Amazon S3 objects at once. To do so, specify a list of the target objects in your manifest and submit that list to Batch

Operations. Your S3 Batch Operations **Object Lock legal hold** job runs until completion, until cancellation, or until a failure state is reached.

S3 Batch Operations verifies that Object Lock is enabled on your S3 bucket before processing any objects in the manifest. To perform the object operations and bucket-level validation, S3 Batch Operations needs the `s3:PutObjectLegalHold` and `s3:GetBucketObjectLockConfiguration` in an AWS Identity and Access Management (IAM) role. These permissions allow S3 Batch Operations to call S3 Object Lock on your behalf.

When you create an S3 Batch Operations job to remove a legal hold, you only need to specify `Off` as the legal hold status. For more information, see the section called “Object Lock considerations”.

For information about how to use this operation with the Amazon S3 REST API, see `S3PutObjectLegalHold` in the CreateJob operation in the *Amazon Simple Storage Service API Reference*.

For an example of using this operation, see Using the AWS SDK for Java.

Restrictions and limitations

When you're using Batch Operations to apply or remove an Object Lock legal hold, the following restrictions and limitations apply:

- S3 Batch Operations doesn't make any bucket-level changes.
- All objects listed in the manifest must be in the same bucket.
- Versioning and S3 Object Lock must be configured on the bucket where the job is performed.
- The **Object Lock legal hold** operation works on the latest version of the object unless a version is explicitly specified in the manifest.
- The `s3:PutObjectLegalHold` permission is required in your IAM role to add or remove a legal hold from objects.
- The `s3:GetBucketObjectLockConfiguration` IAM permission is required to confirm that S3 Object Lock is enabled for the S3 bucket where the job is performed.
- A single S3 Object Lock legal hold job can support a manifest with up to 20 billion objects.

- Copy objects
- Compute checksums
- Delete all object tags

- Invoke AWS Lambda function
- Replace all object tags
- Replace access control list (ACL)
- Restore objects with Batch Operations
- Update object encryption
- Replicating existing objects with Batch Replication
- S3 Object Lock retention
- S3 Object Lock legal hold

Managing S3 Batch Operations jobs

Amazon S3 provides a robust set of tools to help you manage your S3 Batch Operations jobs after you create them. This section describes the operations that you can use to manage and track your jobs by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

Topics

- Using the Amazon S3 console to manage your S3 Batch Operations jobs
- Listing jobs
- Viewing job details
- Assigning job priority

Using the Amazon S3 console to manage your S3 Batch Operations jobs

Using the console, you can manage your S3 Batch Operations jobs. For example, you can:

- View active and queued jobs
- Check the status of a job
- Change a job's priority
- Confirm and run a job
- Clone a job
- Cancel a job

To manage Batch Operations using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. Choose the job that you want to manage.

Listing jobs

You can retrieve a list of your S3 Batch Operations jobs. The list provides information about jobs that haven't yet finished, and jobs that finished within the last 90 days. For each job, the list includes details such as job ID, description, priority, current status, and the number of tasks that have succeeded and failed.

You can filter your job list by status. If you retrieve the list by using the console, you can also search your jobs by description or ID and filter them by AWS Region.

Get a list of `Active` and `Complete` jobs

The following AWS CLI example gets a list of `Active` and `Complete` jobs. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3control list-jobs \
    --region us-west-2 \
    --account-id account-id \
    --job-statuses '["Active","Complete"]' \
    --max-results 20
```

For more information and examples, see list-jobs in the *AWS CLI Command Reference*.

Viewing job details

If you want more information about an Amazon S3 Batch Operations job than you can retrieve by listing jobs, you can view all the details for a single job. You can view details for jobs that haven't yet finished, or jobs that finished within the last 90 days. In addition to the information returned in a job list, a single job's details include information such as:

- The operation parameters.
- Details about the manifest.

- Information about the completion report, if you configured one when you created the job.
- The Amazon Resource Name (ARN) of the AWS Identity and Access Management (IAM) role that you assigned to run the job.

By viewing an individual job's details, you can access a job's entire configuration. To view a job's details, you can use the Amazon S3 console or the AWS Command Line Interface (AWS CLI).

Get an S3 Batch Operations job description in the Amazon S3 console

To view a Batch Operations job description by using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. Choose the job ID of the specific job to view its details.

Get an S3 Batch Operations job description in the AWS CLI

The following example gets the description of an S3 Batch Operations job by using the AWS CLI. To use the following example command, replace the *`user input placeholders`* with your own information.

```
aws s3control describe-job \
--region us-west-2 \
--account-id account-id \
--job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

For more information and examples, see describe-job in the *AWS CLI Command Reference*.

Assigning job priority

You can assign each Amazon S3 Batch Operations job a numeric priority, which can be any positive integer. S3 Batch Operations prioritizes jobs according to the assigned priority. Jobs with a higher priority (or a higher numeric value for the priority parameter) are evaluated first. Priority is determined in descending order. For example, a job queue with a priority value of 10 is given scheduling preference over a job queue with a priority value of 1.

You can change a job's priority while the job is running. If you submit a new job with a higher priority while a job is running, the lower-priority job can pause to allow the higher-priority job to run.

Changing a job's priority doesn't affect the job's processing speed.

> **Note**
>
> S3 Batch Operations honors job priorities on a best-effort basis. Although jobs with higher priorities generally take precedence over jobs with lower priorities, Amazon S3 doesn't guarantee strict ordering of jobs.

Using the S3 console

How to update job priority in the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. Select the specific job that you would like to manage.
4. Choose **Action**. In the dropdown list, choose **Update priority**.

Using the AWS CLI

The following example updates the job priority by using the AWS CLI. A higher number indicates a higher execution priority. To use the following example command, replace the *`user input placeholders`* with your own information.

```
aws s3control update-job-priority \
    --region us-west-2 \
    --account-id account-id \
    --priority 98 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c
```

Using the AWS SDK for Java

To update the priority of an S3 Batch Operations job using the AWS SDK for Java, you can use the S3Control client to modify the job's execution priority, which determines the order in which jobs are processed relative to other jobs in the queue.

For more information about job priority, see Assigning job priority.

For examples of how to update job priority with the AWS SDK for Java, see Update the priority of a batch job in the *Amazon S3 API Reference*.

Tracking job status and completion reports

With S3 Batch Operations, you can view and update job status, add notifications and logging, track job failures, and generate completion reports.

Topics

- Job statuses
- Updating job status
- Notifications and logging
- Tracking job failures
- Completion reports
- Examples: Tracking an S3 Batch Operations job in Amazon EventBridge through AWS CloudTrail
- Examples: S3 Batch Operations completion reports

Job statuses

After you create and run a job, it progresses through a series of statuses. The following table describes the statuses and possible transitions between them.

Status	Description	Transitions
`New`	A job begins in the `New` state when you create it.	A job automatically moves to the `Preparing` state when Amazon S3 begins processing the manifest object.

Status	Description	Transitions
Preparing	Amazon S3 is processing the manifest object and other job parameters to set up and run the job.	A job automatically moves to the Ready state after Amazon S3 finishes processing the manifest and other parameters. The job is then ready to begin running the specified operation on the objects listed in the manifest. If the job requires confirmation before running, such as when you create a job using the Amazon S3 console, then the job transitions from Preparing to Suspended. It remains in the Suspended state until you confirm that you want to run it.
Suspended	The job requires confirmation, but you haven't yet confirmed that you want to run it. Only jobs that you create using the Amazon S3 console require confirmation. A job that's created using the console enters the Suspended state immediately after Preparing. After you confirm that you want to run the job and the job becomes Ready, it never returns to the Suspended state.	After you confirm that you want to run the job, its status changes to Ready.

Status	Description	Transitions
`Ready`	Amazon S3 is ready to begin running the requested object operations.	A job automatically moves to `Active` when Amazon S3 begins to run it. The amount of time that a job remains in the `Ready` state depends on whether you have higher-priority jobs running already and how long those jobs take to complete.
`Active`	Amazon S3 is performing the requested operation on the objects listed in the manifest. While a job is `Active`, you can monitor its progress using the Amazon S3 console or the `DescribeJob` operation through the REST API, AWS CLI, or AWS SDKs.	A job moves out of the `Active` state when the job is no longer running operations on objects. This behavior can happen automatically, such as when a job completes successfully or fails. Or this behavior can occur as a result of user actions, such as canceling a job. The state that the job moves to depends on the reason for the transition.
`Pausing`	The job is transitioning to `Paused` from another state.	A job automatically moves to `Paused` when the `Pausing` stage is finished.
`Paused`	A job can become `Paused` if you submit another job with a higher priority while the current job is running.	A `Paused` job automatically returns to `Active` after any higher-priority jobs that are blocking the job's' execution complete, fail, or are suspended.

Status	Description	Transitions
`Completing`	The job is transitioning to `Complete` from another state.	A job automatically moves to `Complete` when the `Completing` stage is finished.
`Complete`	The job has finished performing the requested operation on all objects in the manifest. The operation might have succeeded or failed for every object. If you configured the job to generate a completion report, the report is available as soon as the job is `Complete`.	`Complete` is a terminal state. Once a job reaches `Complete`, it doesn't transitio n to any other state.
`Cancelling`	The job is transitioning to the `Cancelled` state.	A job automatically moves to `Cancelled` when the `Cancelling` stage is finished.
`Cancelled`	You requested that the job be canceled, and S3 Batch Operations has successfu lly canceled the job. The job won't submit any new requests to Amazon S3.	`Cancelled` is a terminal state. After a job reaches `Cancelled` , the job won't transition to any other state.
`Failing`	The job is transitioning to the `Failed` state.	A job automatically moves to `Failed` once the `Failing` stage is finished.

Status	Description	Transitions
`Failed`	The job has failed and is no longer running. For more information about job failures, see Tracking job failures.	`Failed` is a terminal state. After a job reaches `Failed`, it won't transition to any other state.

Updating job status

The following AWS CLI and AWS SDK for Java examples update the status of a Batch Operations job. For more information about using the Amazon S3 console to manage Batch Operations jobs, see Using the Amazon S3 console to manage your S3 Batch Operations jobs.

Using the AWS CLI

To use the following example commands, replace the *`user input placeholders`* with your own information.

- If you didn't specify the `--no-confirmation-required` parameter in your `create-job` command, the job remains in a suspended state until you confirm the job by setting its status to `Ready`. Amazon S3 then makes the job eligible for execution.

```
aws s3control update-job-status \
    --region us-west-2 \
    --account-id 123456789012 \
    --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
    --requested-job-status 'Ready'
```

- Cancel the job by setting the job status to `Cancelled`.

```
aws s3control update-job-status \
     --region us-west-2 \
     --account-id 123456789012 \
     --job-id 00e123a4-c0d8-41f4-a0eb-b46f9ba5b07c \
     --status-update-reason "No longer needed" \
     --requested-job-status Cancelled
```

Using the AWS SDK for Java

For examples of how to update job status with the AWS SDK for Java, see Update the status of a batch job in the *Amazon S3 API Reference*.

Notifications and logging

In addition to requesting completion reports, you can also capture, review, and audit Batch Operations activity by using AWS CloudTrail. Because Batch Operations uses existing Amazon S3 API operations to perform tasks, those tasks also emit the same events that they would if you called them directly. Therefore, you can track and record the progress of your job and all of its tasks by using the same notification, logging, and auditing tools and processes that you already use with Amazon S3. For more information, see the examples in the following sections.

> **Note**
>
> Batch Operations generates both management and data events in CloudTrail during job execution. The volume of these events scale with the number of keys in each job's manifest. For more information, see the CloudTrail pricing page, which includes examples of how pricing changes depending on the number of trails that you have configured in your account. To learn how to configure and log events to fit your needs, see Create your first trail in the *AWS CloudTrail User Guide*.

For more information about Amazon S3 events, see Amazon S3 Event Notifications.

Tracking job failures

If an S3 Batch Operations job encounters a problem that prevents it from running successfully, such as not being able to read the specified manifest, the job fails. When a job fails, it generates one or more failure codes or failure reasons. S3 Batch Operations stores the failure codes and reasons with the job so that you can view them by requesting the job's details. If you requested a completion report for the job, the failure codes and reasons also appear there.

To prevent jobs from running a large number of unsuccessful operations, Amazon S3 imposes a task-failure threshold on every Batch Operations job. When a job has run at least 1,000 tasks, Amazon S3 monitors the task-failure rate. At any point, if the failure rate (the number of tasks that have failed as a proportion of the total number of tasks that have run) exceeds 50 percent, the job

fails. If your job fails because it exceeded the task-failure threshold, you can identify the cause of the failures. For example, you might have accidentally included some objects in the manifest that don't exist in the specified bucket. After fixing the errors, you can resubmit the job.

> **Note**
>
> S3 Batch Operations operates asynchronously and the tasks don't necessarily run in the order that the objects are listed in the manifest. Therefore, you can't use the manifest ordering to determine which objects' tasks succeeded and which ones failed. Instead, you can examine the job's completion report (if you requested one) or view your AWS CloudTrail event logs to help determine the source of the failures.

Completion reports

When you create a job, you can request a completion report. As long as S3 Batch Operations successfully invokes at least one task, Amazon S3 generates a completion report after the job finishes running tasks, fails, or is canceled. You can configure the completion report to include all tasks or only failed tasks.

The completion report includes the job configuration, status, and information for each task, including the object key and version, status, error codes, and descriptions of any errors. Completion reports provide an easy way to view the results of your tasks in a consolidated format with no additional setup required. Completion reports are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3). For an example of a completion report, see Examples: S3 Batch Operations completion reports.

If you don't configure a completion report, you can still monitor and audit your job and its tasks by using CloudTrail and Amazon CloudWatch. For more information, see the following topics:

Topics

- Examples: Tracking an S3 Batch Operations job in Amazon EventBridge through AWS CloudTrail
- Examples: S3 Batch Operations completion reports

Examples: Tracking an S3 Batch Operations job in Amazon EventBridge through AWS CloudTrail

Amazon S3 Batch Operations job activity is recorded as events in AWS CloudTrail. You can create a custom rule in Amazon EventBridge and send these events to the target notification resource of your choice, such as Amazon Simple Notification Service (Amazon SNS).

> **Note**
>
> Amazon EventBridge is the preferred way to manage your events. Amazon CloudWatch Events and EventBridge are the same underlying service and API, but EventBridge provides more features. Changes that you make in either CloudWatch or EventBridge appear in each console. For more information, see the *Amazon EventBridge User Guide*.

Tracking Examples

- S3 Batch Operations events recorded in CloudTrail
- EventBridge rule for tracking S3 Batch Operations job events

S3 Batch Operations events recorded in CloudTrail

When a Batch Operations job is created, it is recorded as a `JobCreated` event in CloudTrail. As the job runs, it changes state during processing, and other `JobStatusChanged` events are recorded in CloudTrail. You can view these events on the CloudTrail console. For more information about CloudTrail, see the *AWS CloudTrail User Guide*.

> **Note**
>
> Only S3 Batch Operations job `status-change` events are recorded in CloudTrail.

Example— S3 Batch Operations job completion event recorded by CloudTrail

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "accountId": "123456789012",
```

```
        "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-05T18:25:30Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "JobStatusChanged",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "requestParameters": null,
    "responseElements": null,
    "eventID": "f907577b-bf3d-4c53-b9ed-8a83a118a554",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "recipientAccountId": "123412341234",
    "serviceEventDetails": {
        "jobId": "d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "jobArn": "arn:aws:s3:us-west-2:181572960644:job/
d6e58ec4-897a-4b6d-975f-10d7f0fb63ce",
        "status": "Complete",
        "jobEventId": "b268784cf0a66749f1a05bce259804f5",
        "failureCodes": [],
        "statusChangeReason": []
    }
}
```

EventBridge rule for tracking S3 Batch Operations job events

The following example shows how to create a rule in Amazon EventBridge to capture S3 Batch Operations events recorded by AWS CloudTrail to a target of your choice.

To do this, you create a rule by following all the steps in Creating EventBridge rules that react to events. You paste the following S3 Batch Operations custom event pattern policy where applicable, and choose the target service of your choice.

S3 Batch Operations custom event pattern policy

```
{
    "source": [
        "aws.s3"
    ],
    "detail-type": [
        "AWS Service Event via CloudTrail"
    ],
```

```
    "detail": {
        "eventSource": [
            "s3.amazonaws.com"
        ],
        "eventName": [
            "JobCreated",
            "JobStatusChanged"
        ]
    }
}
```

The following examples are two Batch Operations events that were sent to Amazon Simple Queue Service (Amazon SQS) from an EventBridge event rule. A Batch Operations job goes through many different states while processing (`New`, `Preparing`, `Active`, etc.), so you can expect to receive several messages for each job.

Example— JobCreated sample event

```
{
    "version": "0",
    "id": "51dc8145-541c-5518-2349-56d7dffdf2d8",
    "detail-type": "AWS Service Event via CloudTrail",
    "source": "aws.s3",
    "account": "123456789012",
    "time": "2020-02-27T15:25:49Z",
    "region": "us-east-1",
    "resources": [],
    "detail": {
        "eventVersion": "1.05",
        "userIdentity": {
            "accountId": "11112223334444",
            "invokedBy": "s3.amazonaws.com"
        },
        "eventTime": "2020-02-27T15:25:49Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "JobCreated",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "s3.amazonaws.com",
        "userAgent": "s3.amazonaws.com",
        "eventID": "7c38220f-f80b-4239-8b78-2ed867b7d3fa",
        "readOnly": false,
        "eventType": "AwsServiceEvent",
        "serviceEventDetails": {
```

```
            "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
            "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
            "status": "New",
            "jobEventId": "f177ff24f1f097b69768e327038f30ac",
            "failureCodes": [],
            "statusChangeReason": []
        }
    }
}
```

Example— JobStatusChanged job completion event

```
{
  "version": "0",
  "id": "c8791abf-2af8-c754-0435-fd869ce25233",
  "detail-type": "AWS Service Event via CloudTrail",
  "source": "aws.s3",
  "account": "123456789012",
  "time": "2020-02-27T15:26:42Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "eventVersion": "1.05",
    "userIdentity": {
      "accountId": "1111222233334444",
      "invokedBy": "s3.amazonaws.com"
    },
    "eventTime": "2020-02-27T15:26:42Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "JobStatusChanged",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "s3.amazonaws.com",
    "userAgent": "s3.amazonaws.com",
    "eventID": "0238c1f7-c2b0-440b-8dbd-1ed5e5833afb",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "serviceEventDetails": {
      "jobId": "e849b567-5232-44be-9a0c-40988f14e80c",
      "jobArn": "arn:aws:s3:us-east-1:181572960644:job/
e849b567-5232-44be-9a0c-40988f14e80c",
      "status": "Complete",
      "jobEventId": "51f5ac17dba408301d56cd1b2c8d1e9e",
```

```
      "failureCodes": [],
      "statusChangeReason": []
    }
  }
}
```

Examples: S3 Batch Operations completion reports

When you create an S3 Batch Operations job, you can request a completion report for all tasks or just for failed tasks. As long as at least one task has been invoked successfully, S3 Batch Operations generates a report for jobs that have completed, failed, or been canceled.

The completion report contains additional information for each task, including the object key name and version, status, error codes, and descriptions of any errors. The description of errors for each failed task can be used to diagnose issues that occur during job creation, such as permissions. For **Compute checksum** jobs, the completion report contains checksum values for every object.

> **Note**
>
> Completion reports are always encrypted with Amazon S3 managed keys (SSE-S3).

Example— Top-level manifest result file

The top-level `manifest.json` file contains the locations of each succeeded report and (if the job had any failures) the location of failed reports, as shown in the following example.

```
{
    "Format": "Report_CSV_20180820",
    "ReportCreationDate": "2019-04-05T17:48:39.725Z",
    "Results": [
        {
            "TaskExecutionStatus": "succeeded",
            "Bucket": "my-job-reports",
            "MD5Checksum": "83b1c4cbe93fc893f54053697e10fd6e",
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/
results/6217b0fab0de85c408b4be96aeaca9b195a7daa5.csv"
        },
        {
            "TaskExecutionStatus": "failed",
            "Bucket": "my-job-reports",
```

```
            "MD5Checksum": "22ee037f3515975f7719699e5c416eaa",
            "Key": "job-f8fb9d89-a3aa-461d-bddc-ea6a1b131955/results/
b2ddad417e94331e9f37b44f1faf8c7ed5873f2e.csv"
        }
    ],
    "ReportSchema": "Bucket, Key, VersionId, TaskStatus, ErrorCode, HTTPStatusCode,
 ResultMessage"
}
```

Succeeded task reports

Succeeded tasks reports contain the following for the *successful* tasks:

- `Bucket`
- `Key`
- `VersionId`
- `TaskStatus`
- `ErrorCode`
- `HTTPStatusCode`
- `ResultMessage`

Failed task reports

Failed task reports contain the following information for all *failed* tasks:

- `Bucket`
- `Key`
- `VersionId`
- `TaskStatus`
- `ErrorCode`
- `HTTPStatusCode`
- `ResultMessage`

Example— Lambda function task report

In the following example, the Lambda function successfully copied the Amazon S3 object to another bucket. The returned Amazon S3 response is passed back to S3 Batch Operations and is then written into the final completion report.

```
amzn-s3-demo-bucket1,image_17775,,succeeded,200,,"{u'CopySourceVersionId':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'RequestId': '3ED5852152014362', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'nXNaClIMxEJzWNmeMNQV2KpjbaCJLn0OGoXWZpuVOFS/
iQYWxb3QtTvzX9SVfx2lA3oTKLwImKw=', 'x-amz-copy-source-version-id':
 'xVR78haVKlRnurYofbTfYr3ufYbktF8h', 'server': 'AmazonS3', 'x-amz-request-id':
 '3ED5852152014362', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT', 'content-type':
 'application/xml'}}}"
amzn-s3-demo-bucket1,image_17763,,succeeded,200,,"{u'CopySourceVersionId':
 '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 39, tzinfo=tzlocal()),
 u'ETag': '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata':
 {'HTTPStatusCode': 200, 'RetryAttempts': 0, 'HostId': 'GiCZNYr8LHd/
Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'RequestId':
 '1BC9F5B1B95D7000', 'HTTPHeaders': {'content-length': '234', 'x-amz-id-2':
 'GiCZNYr8LHd/Thyk6beTRP96IGZk2sYxujLe13TuuLpq6U2RD3we0YoluuIdm1PRvkMwnEW1aFc=', 'x-
amz-copy-source-version-id': '6HjOUSim4Wj6BTcbxToXW44pSZ.40pwq', 'server': 'AmazonS3',
 'x-amz-request-id': '1BC9F5B1B95D7000', 'date': 'Fri, 05 Apr 2019 17:35:39 GMT',
 'content-type': 'application/xml'}}}"
amzn-s3-demo-bucket1,image_17860,,succeeded,200,,"{u'CopySourceVersionId':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', u'CopyObjectResult': {u'LastModified':
 datetime.datetime(2019, 4, 5, 17, 35, 40, tzinfo=tzlocal()), u'ETag':
 '""fe66f4390c50f29798f040d7aae72784""'}, 'ResponseMetadata': {'HTTPStatusCode':
 200, 'RetryAttempts': 0, 'HostId': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'RequestId': '8D9CA56A56813DF3', 'HTTPHeaders':
 {'content-length': '234', 'x-amz-id-2': 'F9ooZOgpE5g9sNgBZxjdiPHqB4+0DNWgj3qbsir
+sKai4fv7rQEcF2fBN1VeeFc2WH45a9ygb2g=', 'x-amz-copy-source-version-id':
 'm.MDD0g_QsUnYZ8TBzVFrp.TmjN8PJyX', 'server': 'AmazonS3', 'x-amz-request-id':
 '8D9CA56A56813DF3', 'date': 'Fri, 05 Apr 2019 17:35:40 GMT', 'content-type':
 'application/xml'}}}"
```

The following example report shows a case in which the AWS Lambda function timed out, causing failures to exceed the failure threshold. It is then marked as a `PermanentFailure`.

```
amzn-s3-demo-bucket1,image_14975,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:21.155Z 2845ca0d-38d9-4c4b-
abcf-379dc749c452 Task timed out after 3.00 seconds""}"
amzn-s3-demo-bucket1,image_15897,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:29.610Z 2d0a330b-de9b-425f-
b511-29232fde5fe4 Task timed out after 3.00 seconds""}"
amzn-s3-demo-bucket1,image_14819,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:22.362Z fcf5efde-74d4-4e6d-b37a-
c7f18827f551 Task timed out after 3.00 seconds""}"
amzn-s3-demo-bucket1,image_15930,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:29.809Z 3dd5b57c-4a4a-48aa-8a35-
cbf027b7957e Task timed out after 3.00 seconds""}"
amzn-s3-demo-bucket1,image_17644,,failed,200,PermanentFailure,"Lambda
 returned function error: {""errorMessage"":""2019-04-05T17:35:46.025Z
 10a764e4-2b26-4d8c-9056-1e1072b4723f Task timed out after 3.00 seconds""}"
amzn-s3-demo-bucket1,image_17398,,failed,200,PermanentFailure,"Lambda returned
 function error: {""errorMessage"":""2019-04-05T17:35:44.661Z 1e306352-4c54-4eba-
aee8-4d02f8c0235c Task timed out after 3.00 seconds""}"
```

Example— Compute checksum task report

In the following example, the **Compute checksum** operation successfully calculated the checksum for the uploaded object while at rest. The returned Amazon S3 response is passed back to S3 Batch Operations, and is then written into the final completion report:

```
amzn-s3-demo-bucket1,s3-standard-1mb-test-
object,,succeeded,200,,"{""checksum_base64"":""bS9TOQ\u003d
\u003d"",""etag"":""3c3c18130429890945 98e4b57ecbdc82"",""checksumAlgorithm"":""CRC32"",""checks
```

The following example report shows what happens when a **Compute checksum** operation fails, resulting in a failed task report:

```
amzn-s3-demo-bucket1,image_14975,,failed,200,PermanentFailure,"error details:
 {""failureMessage"":"Task 2845ca0d-38d9-4c4b-abcf-379dc749c452 SSE-C encryption type
 is not supported for this operation", ""errorCode"": "400"}"
amzn-s3-demo-bucket1,image_14975,,failed,200,PermanentFailure,"error details:
 {""failureMessage"":"Task 2845ca0d-38d9-4c4b-abcf-379dc749c452 Key not found",
 ""errorCode"": "404"}"
amzn-s3-demo-bucket1,image_14975,,failed,200,PermanentFailure,"error details:
 {""failureMessage"":"Task 2845ca0d-38d9-4c4b-abcf-379dc749c452 Internal server error,
 please retry", ""errorCode"": "500"}"
```

Controlling access and labeling jobs using tags

You can label and control access to your S3 Batch Operations jobs by adding *tags*. Tags can be used to identify who is responsible for a Batch Operations job. The presence of job tags can grant or limit a user's ability to cancel a job, activate a job in the confirmation state, or change a job's priority level. You can create jobs with tags attached to them, and you can add tags to jobs after they're created. Each tag is a key-value pair that can be included when you create the job or updated later.

> **Warning**
>
> Make sure that your job tags don't contain any confidential information or personal data.

Consider the following tagging example: Suppose that you want your Finance department to create a Batch Operations job. You could write an AWS Identity and Access Management (IAM) policy that allows a user to invoke `CreateJob`, provided that the job is created with the `Department` tag assigned the value `Finance`. Furthermore, you could attach that policy to all users who are members of the Finance department.

Continuing with this example, you could write a policy that allows a user to update the priority of any job that has the desired tags, or cancel any job that has those tags. For more information, see the section called “Controlling permissions”.

You can add tags to new S3 Batch Operations jobs when you create them, or you can add them to existing jobs.

Note the following tag restrictions:

- You can associate up to 50 tags with a job as long as they have unique tag keys.
- A tag key can be up to 128 Unicode characters in length, and tag values can be up to 256 Unicode characters in length.
- The key and values are case sensitive.

For more information about tag restrictions, see User-Defined Tag Restrictions in the *AWS Billing and Cost Management User Guide*.

API operations related to S3 Batch Operations job tagging

Amazon S3 supports the following API operations that are specific to S3 Batch Operations job tagging:

- GetJobTagging – Returns the tag set associated with a Batch Operations job.
- PutJobTagging – Replaces the tag set associated with a job. There are two distinct scenarios for S3 Batch Operations job tag management using this API action:
  - Job has no tags – You can add a set of tags to a job (the job has no prior tags).
  - Job has a set of existing tags – To modify the existing tag set, you can either replace the existing tag set entirely, or make changes within the existing tag set by retrieving the existing tag set using GetJobTagging, modify that tag set, and use this API action to replace the tag set with the one you have modified.

    > **Note**
    >
    > If you send this request with an empty tag set, S3 Batch Operations deletes the existing tag set on the object. If you use this method, you are charged for a Tier 1 Request (PUT). For more information, see Amazon S3 pricing.
    > To delete existing tags for your Batch Operations job, the `DeleteJobTagging` action is preferred because it achieves the same result without incurring charges.
- DeleteJobTagging – Deletes the tag set associated with a Batch Operations job.

Creating a Batch Operations job with job tags used for labeling

You can label and control access to your Amazon S3 Batch Operations jobs by adding *tags*. Tags can be used to identify who is responsible for a Batch Operations job. You can create jobs with tags attached to them, and you can add tags to jobs after they are created. For more information, see the section called "Using tags".

Using the AWS CLI

The following AWS CLI example creates an S3 Batch Operations `S3PutObjectCopy` job using job tags as labels for the job.

1. Select the action or `OPERATION` that you want the Batch Operations job to perform, and choose your `TargetResource`.

```
read -d '' OPERATION <<EOF
{
  "S3PutObjectCopy": {
    "TargetResource": "arn:aws:s3:::amzn-s3-demo-destination-bucket"
  }
}
EOF
```

2. Identify the job TAGS that you want for the job. In this case, you apply two tags, `department` and `FiscalYear`, with the values `Marketing` and `2020` respectively.

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
]
EOF
```

3. Specify the `MANIFEST` for the Batch Operations job.

```
read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "EXAMPLE_S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/example_manifest.csv",
    "ETag": "example-5dc7a8bfb90808fc5d546218"
  }
}
EOF
```

4. Configure the REPORT for the Batch Operations job.

```
read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
  "Format": "Example_Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/copy-with-replace-metadata",
  "ReportScope": "AllTasks"
}
EOF
```

5. Run the`create-job` action to create your Batch Operations job with inputs set in the preceding steps.

```
aws \
    s3control create-job \
    --account-id 123456789012 \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn arn:aws:iam::123456789012:role/batch-operations-role \
    --tags "${TAGS//$'\n'/}" \
    --client-request-token "$(uuidgen)" \
    --region us-west-2 \
    --description "Copy with Replace Metadata";
```

Using the AWS SDK for Java

To create an S3 Batch Operations job with tags using the AWS SDK for Java, you can use the S3Control client to configure job parameters including manifest location, job operations, reporting settings, and tags for organization and tracking purposes.

For examples of how to create S3 Batch Operations jobs with tags using the AWS SDK for Java, see Create a batch job to copy objects in the *Amazon S3 API Reference*.

Deleting the tags from an S3 Batch Operations job

You can use these examples to delete the tags from an Amazon S3 Batch Operations job.

Using the AWS CLI

The following example deletes the tags from a Batch Operations job using the AWS CLI.

```
aws \
    s3control delete-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1
```

Delete the job tags of a Batch Operations job

To delete the tags of an S3 Batch Operations job using the AWS SDK for Java, you can use the S3Control client with the job ID to remove all tags associated with the batch operations job.

For examples of how to delete job tags with the AWS SDK for Java, see Delete tags from a batch job in the *Amazon S3 API Reference*.

Adding job tags to an existing Batch Operations job

You can use the PutJobTagging API operation to add job tags to your existing Amazon S3 Batch Operations jobs. For more information, see the following examples.

Using the AWS CLI

The following is an example of using `s3control put-job-tagging` to add job tags to your S3 Batch Operations job by using the AWS CLI. To use the examples, replace the *`user input placeholders`* with your own information.

> **Note**
>
> If you send this request with an empty tag set, Batch Operations deletes the existing tag set on the object. However, if you use this approach, you are charged for a Tier 1 Request (PUT). For more information, see Amazon S3 pricing.
> Instead, to delete existing tags for your Batch Operations job, we recommend using the `DeleteJobTagging` operation because it achieves the same result without incurring charges.

1. Identify the job TAGS that you want for the job. In this case, you apply two tags, *`department`* and *`FiscalYear`*, with the values *`Marketing`* and *`2020`* respectively.

```
read -d '' TAGS <<EOF
[
  {
    "Key": "department",
    "Value": "Marketing"
  },
  {
    "Key": "FiscalYear",
    "Value": "2020"
  }
]
EOF
```

2. Run the following `put-job-tagging` command with the required parameters:

```
aws \
    s3control put-job-tagging \
    --account-id 123456789012 \
    --tags "${TAGS//$'\n'/}" \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1
```

Using the AWS SDK for Java

To put tags on an S3 Batch Operations job using the AWS SDK for Java, you can use the S3Control client to add or update tags with key-value pairs for organization and tracking purposes.

For examples of how to put job tags with the AWS SDK for Java, see Add tags to a batch job in the *Amazon S3 API Reference*.

Getting the tags of a S3 Batch Operations job

To retrieve the tags of an Amazon S3 Batch Operations job, you can use the `GetJobTagging` API operation. For more information, see the following examples.

Using the AWS CLI

The following example gets the tags of a Batch Operations job using the AWS CLI. To use this example, replace the *`user input placeholders`* with your own information.

```
aws \
```

```
    s3control get-job-tagging \
    --account-id 123456789012 \
    --job-id Example-e25a-4ed2-8bee-7f8ed7fc2f1c \
    --region us-east-1
```

Using the AWS SDK for Java

To get the tags of an S3 Batch Operations job using the AWS SDK for Java, you can use the S3Control client with the job ID to retrieve all tags associated with the batch operations job and return them as a list.

For examples of how to get job tags with the AWS SDK for Java, see Get tags from a batch job in the *Amazon S3 API Reference*.

Controlling permissions for Batch Operations using job tags

To help you manage your Amazon S3 Batch Operations jobs, you can add *job tags*. With job tags, you can control access to your Batch Operations jobs and enforce that tags be applied when any job is created.

You can apply up to 50 job tags to each Batch Operations job. By using tags, you can set granular policies to restrict the set of users that can edit the job. Job tags can grant or limit a user's ability to cancel a job, activate a job in the confirmation state, or change a job's priority level. In addition, you can enforce that tags be applied to all new jobs, and specify the allowed key-value pairs for the tags. You can express all of these conditions by using AWS Identity and Access Management (IAM) policy language. For more information, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

The following example shows how you can use S3 Batch Operations job tags to grant users permission to create and edit only the jobs that are run within a specific department (for example, the Finance or Compliance department). You can also assign jobs based on the stage of development that they are related to, such as QA or Production.

In this example, you use S3 Batch Operations job tags in IAM policies to grant users permission to create and edit only the jobs that are being run within their department. You assign jobs based on the stage of development that they are related to, such as QA or Production.

The following examples use the following departments, with each department using Batch Operations in different ways:

- Finance
- Compliance
- Business Intelligence
- Engineering

Topics

- Controlling access by assigning tags to users and resources
- Tagging Batch Operations jobs by stage and enforcing limits on job priority

Controlling access by assigning tags to users and resources

In this scenario, the administrators are using attribute-based access control (ABAC). ABAC is an IAM authorization strategy that defines permissions by attaching tags to both users and AWS resources.

Users and jobs are assigned one of the following department tags:

Key : Value

- `department : Finance`
- `department : Compliance`
- `department : BusinessIntelligence`
- `department : Engineering`

> **Note**
>
> Job tag keys and values are case sensitive.

Using the ABAC access control strategy, you grant a user in the Finance department permission to create and manage S3 Batch Operations jobs within their department by associating the tag `department=Finance` with their user.

Furthermore, you can attach a managed policy to the IAM user that allows any user in their company to create or modify S3 Batch Operations jobs within their respective departments.

The policy in this example includes three policy statements:

- The first statement in the policy allows the user to create a Batch Operations job provided that the job creation request includes a job tag that matches their respective department. This is expressed using the `"${aws:PrincipalTag/department}"` syntax, which is replaced by the user's department tag at policy evaluation time. The condition is satisfied when the value provided for the department tag in the request (`"aws:RequestTag/department"`) matches the user's department.
- The second statement in the policy allows users to change the priority of jobs or update a job's status provided that the job the user is updating matches the user's department.
- The third statement allows a user to update a Batch Operations job's tags at any time through a `PutJobTagging` request as long as (1) their department tag is preserved and (2) the job they're updating is within their department.

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [
            {
                  "Effect": "Allow",
                  "Action": "s3:CreateJob",
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:RequestTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": [
                        "s3:UpdateJobPriority",
                        "s3:UpdateJobStatus"
            ],
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
```

```
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
            {
                  "Effect": "Allow",
                  "Action": "s3:PutJobTagging",
                  "Resource": "*",
                  "Condition": {
                        "StringEquals": {
                              "aws:RequestTag/department": "${aws:PrincipalTag/
department}",
                              "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        }
    ]
}
```

Tagging Batch Operations jobs by stage and enforcing limits on job priority

All S3 Batch Operations jobs have a numeric priority, which Amazon S3 uses to decide in what order to run the jobs. For this example, you restrict the maximum priority that most users can assign to jobs, with higher priority ranges reserved for a limited set of privileged users, as follows:

- QA stage priority range (low): 1-100
- Production stage priority range (high): 1-300

To do this, introduce a new tag set representing the stage of the job:

Key : Value

- `stage : QA`
- `stage : Production`

Creating and updating low-priority jobs within a department

This policy introduces two new restrictions on S3 Batch Operations job creation and update, in addition to the department-based restriction:

- It allows users to create or update jobs in their department with a new condition that requires the job to include the tag `stage=QA`.
- It allows users to create or update a job's priority up to a new maximum priority of 100.

```
{
        "Version": "2012-10-17",
        "Statement": [
        {
        "Effect": "Allow",
        "Action": "s3:CreateJob",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department": "${aws:PrincipalTag/department}",
                "aws:RequestTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": [
            "s3:UpdateJobStatus"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:UpdateJobPriority",
        "Resource": "*",
```

```
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
                "aws:ResourceTag/stage": "QA"
            },
            "NumericLessThanEquals": {
                "s3:RequestJobPriority": 100
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:PutJobTagging",
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/department" : "${aws:PrincipalTag/department}",
                "aws:ResourceTag/department": "${aws:PrincipalTag/department}",
                "aws:RequestTag/stage": "QA",
                "aws:ResourceTag/stage": "QA"
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:GetJobTagging",
        "Resource": "*"
    }
    ]
}
```

Creating and updating high-priority jobs within a department

A small number of users might require the ability to create high priority jobs in either QA or Production. To support this need, you create a managed policy that's adapted from the low-priority policy in the previous section.

This policy does the following:

- Allows users to create or update jobs in their department with either the tag `stage=QA` or `stage=Production`.
- Allows users to create or update a job's priority up to a maximum of 300.

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [
          {
                "Effect": "Allow",
                "Action": "s3:CreateJob",
                "Resource": "*",
                "Condition": {
                      "ForAnyValue:StringEquals": {
                            "aws:RequestTag/stage": [
                                  "QA",
                                  "Production"
                    ]
                },
                      "StringEquals": {
                            "aws:RequestTag/department": "${aws:PrincipalTag/
department}"
                },
                      "NumericLessThanEquals": {
                            "s3:RequestJobPriority": 300
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": [
                      "s3:UpdateJobStatus"
            ],
                "Resource": "*",
                "Condition": {
                      "StringEquals": {
                            "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": "s3:UpdateJobPriority",
                "Resource": "*",
                "Condition": {
```

```
                    "ForAnyValue:StringEquals": {
                        "aws:ResourceTag/stage": [
                            "QA",
                            "Production"
                    ]
                },
                    "StringEquals": {
                        "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                },
                    "NumericLessThanEquals": {
                        "s3:RequestJobPriority": 300
                }
            }
        },
          {
                "Effect": "Allow",
                "Action": "s3:PutJobTagging",
                "Resource": "*",
                "Condition": {
                    "StringEquals": {
                        "aws:RequestTag/department": "${aws:PrincipalTag/
department}",
                        "aws:ResourceTag/department": "${aws:PrincipalTag/
department}"
                },
                    "ForAnyValue:StringEquals": {
                        "aws:RequestTag/stage": [
                            "QA",
                            "Production"
                    ],
                        "aws:ResourceTag/stage": [
                            "QA",
                            "Production"
                    ]
                }
            }
        }
    ]
}
```

Managing S3 Object Lock using S3 Batch Operations

You can use S3 Batch Operations to perform large-scale batch operations on Amazon S3 objects. S3 Batch Operations can perform a single operation on lists of Amazon S3 objects that you specify. A single job can perform a specified operation on billions of objects containing exabytes of data. Amazon S3 tracks progress, sends notifications, and stores a detailed completion report of all actions, providing a fully managed, auditable, and serverless experience. You can use S3 Batch Operations through the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API.

With S3 Object Lock, you can place a legal hold on an object version. Like setting a retention period, a legal hold prevents an object version from being overwritten or deleted. However, a legal hold doesn't have an associated retention period and remains in effect until the legal hold is removed. For more information, see S3 Object Lock legal hold.

To use S3 Batch Operations with Object Lock to add legal holds to many Amazon S3 objects at once, see the following topics.

Topics

- Enabling S3 Object Lock using S3 Batch Operations
- Setting Object Lock retention using Batch Operations
- Using S3 Batch Operations with S3 Object Lock retention compliance mode
- Use S3 Batch Operations with S3 Object Lock retention governance mode
- Using S3 Batch Operations to turn off S3 Object Lock legal holds

Enabling S3 Object Lock using S3 Batch Operations

You can use Amazon S3 Batch Operations with S3 Object Lock to manage retention or enable a legal hold for many Amazon S3 objects at once. You specify the list of target objects in your manifest and submit it to Batch Operations for completion. For more information, see the section called “Object Lock retention” and the section called “Object Lock legal hold”.

The following examples show how to create an AWS Identity and Access Management (IAM) role with S3 Batch Operations permissions and update the role permissions to create jobs that enable Object Lock. You must also have a CSV manifest that identifies the objects for your S3 Batch Operations job. For more information, see the section called “Specifying a manifest”.

To use the following examples, replace the *`user input placeholders`* with your own information.

Using the AWS CLI

1. Create an IAM role and assign S3 Batch Operations permissions to run.

   This step is required for all S3 Batch Operations jobs.

```
export AWS_PROFILE='aws-user'

read -d '' batch_operations_trust_policy <<EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "batchoperations.s3.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
aws iam create-role --role-name batch_operations-objectlock \
--assume-role-policy-document "${batch_operations_trust_policy}"
```

2. Set up S3 Batch Operations with S3 Object Lock to run.

   In this step, you allow the role to do the following:

   a. Run Object Lock on the S3 bucket that contains the target objects that you want Batch Operations to run on.

   b. Read the S3 bucket where the manifest CSV file and the objects are located.

   c. Write the results of the S3 Batch Operations job to the reporting bucket.

```
read -d '' batch_operations_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": "s3:GetBucketObjectLockConfiguration",
            "Resource": [
                "arn:aws:s3:::{{amzn-s3-demo-manifest-bucket}}"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{amzn-s3-demo-manifest-bucket}}/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "arn:aws:s3:::{{amzn-s3-demo-completion-report-bucket}}/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name batch_operations-objectlock \
--policy-name object-lock-permissions \
--policy-document "${batch_operations_permissions}"
```

Using the AWS SDK for Java

You can create an IAM role with S3 Batch Operations permissions, and update the role permissions to create jobs that enable Object Lock by using the AWS SDK for Java. You must also have a CSV manifest identifying the objects for your S3 Batch Operations job. For more information, see the section called “Specifying a manifest”.

Perform the following steps:

1. Create an IAM role and assign S3 Batch Operations permissions to run. This step is required for all S3 Batch Operations jobs.
2. Set up S3 Batch Operations with S3 Object Lock to run.

   You allow the role to do the following:

   1. Run Object Lock on the S3 bucket that contains the target objects that you want Batch Operations to run on.
   2. Read the S3 bucket where the manifest CSV file and the objects are located.
   3. Write the results of the S3 Batch Operations job to the reporting bucket.

For a code example that demonstrates how to create an IAM role for enabling S3 Object Lock using S3 Batch Operations with the AWS SDK for Java, see CreateObjectLockRole.java in the *AWS SDK for Java 2.x Code Examples*.

Setting Object Lock retention using Batch Operations

You can use Amazon S3 Batch Operations with S3 Object Lock to manage retention for many Amazon S3 objects at once. You specify the list of target objects in your manifest and submit it to Batch Operations for completion. For more information, see the section called "Object Lock retention" and the section called "Object Lock legal hold".

The following examples show how to create an AWS Identity and Access Management (IAM) role with S3 Batch Operations permissions and update the role permissions to include the `s3:PutObjectRetention` permissions so that you can run S3 Object Lock retention on the objects in your manifest bucket. You must also have a CSV manifest that identifies the objects for your S3 Batch Operations job. For more information, see the section called "Specifying a manifest".

To use the following examples, replace the *`user input placeholders`* with your own information.

Using the AWS CLI

The following AWS CLI example shows how to use Batch Operations to apply S3 Object Lock retention across multiple objects.

```
export AWS_PROFILE='aws-user'
```

```
read -d '' retention_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": [
                "arn:aws:s3:::{{amzn-s3-demo-manifest-bucket}}/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name batch_operations-objectlock --policy-
name retention-permissions --policy-document "${retention_permissions}"
```

Using the AWS SDK for Java

For examples of how to use Batch Operations to apply S3 Object Lock retention across multiple objects with the AWS SDK for Java, see Use CreateJob with an AWS SDK or CLI in the *Amazon S3 API Reference*.

Using S3 Batch Operations with S3 Object Lock retention compliance mode

The following example builds on the previous examples of creating a trust policy and setting S3 Batch Operations and S3 Object Lock configuration permissions on your objects. This example sets the retention mode to `COMPLIANCE` and the `retain until date` to January 1, 2025. This example creates a job that targets objects in the manifest bucket and reports the results in the reports bucket that you identified.

To use the following examples, replace the *`user input placeholders`* with your own information.

Using the AWS CLI

The following AWS CLI examples show how to use Batch Operations to apply S3 Object Lock retention compliance mode across multiple objects.

Example— Set S3 Object Lock retention compliance mode across multiple objects

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/batch_operations-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-01T00:00:00",
      "Mode":"COMPLIANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/compliance-objects-
manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::ReportBucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "amzn-s3-demo-completion-report-bucket/compliance-objects-batch-
operations",
  "ReportScope": "AllTasks"
}
```

```
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Set compliance retain-until to 1 Jul 2030";
```

Example— Extend the COMPLIANCE mode's retain until date to January 15, 2025

The following example extends the COMPLIANCE mode's `retain until date` to January 15, 2025.

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/batch_operations-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-15T00:00:00",
      "Mode":"COMPLIANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
```

```
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/compliance-objects-
manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/compliance-objects-batch_operations",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Extend compliance retention to 15 Jan 2025";
```

Using the AWS SDK for Java

The following AWS SDK for Java examples show how to use Batch Operations to apply S3 Object Lock retention compliance mode across multiple objects, including setting the retention mode to COMPLIANCE with a retain until date and extending the COMPLIANCE mode retain until date.

For examples of how to use Batch Operations to apply S3 Object Lock retention compliance mode across multiple objects with the AWS SDK for Java, see Use CreateJob with an AWS SDK or CLI in the *Amazon S3 API Reference*.

Use S3 Batch Operations with S3 Object Lock retention governance mode

The following example builds on the previous example of creating a trust policy, and setting S3 Batch Operations and S3 Object Lock configuration permissions. This example shows how to apply S3 Object Lock retention governance with the `retain until date` of January 30, 2025, across multiple objects. It creates a Batch Operations job that uses the manifest bucket and reports the results in the reports bucket.

To use the following examples, replace the *`user input placeholders`* with your own information.

Using the AWS CLI

The following AWS CLI examples show how to use Batch Operations to apply S3 Object Lock retention governance mode across multiple objects.

Example— Apply S3 Object Lock retention governance across multiple objects with the retain until date of January 30, 2025

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/batch_operations-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "Retention": {
      "RetainUntilDate":"2025-01-30T00:00:00",
      "Mode":"GOVERNANCE"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
```

```
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/governance-objects-
manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::amzn-s3-demo-completion-report-bucketT",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/governance-objects",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Put governance retention";
```

Example— Bypass retention governance across multiple objects

The following example builds on the previous example of creating a trust policy, and setting S3 Batch Operations and S3 Object Lock configuration permissions. It shows how to bypass retention governance across multiple objects and creates a Batch Operations job that uses the manifest bucket and reports the results in the reports bucket.

```
export AWS_PROFILE='aws-user'

read -d '' bypass_governance_permissions <<EOF
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:BypassGovernanceRetention"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        }
    ]
}
EOF

aws iam put-role-policy --role-name batch-operations-objectlock --policy-name bypass-
governance-permissions --policy-document "${bypass_governance_permissions}"

export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
export ROLE_ARN='arn:aws:iam::123456789012:role/batch_operations-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectRetention": {
    "BypassGovernanceRetention": true,
    "Retention": {
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/governance-objects-
manifest.csv",
```

```
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/batch_operations-governance",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Remove governance retention";
```

Using the AWS SDK for Java

The following AWS SDK for Java examples show how to apply S3 Object Lock retention governance with the `retain until date` set to January 30, 2025 across multiple objects, including applying Object Lock retention governance across multiple objects with a retain until date and bypassing retention governance across multiple objects.

For examples of how to use Batch Operations with S3 Object Lock retention governance mode with the AWS SDK for Java, see Use CreateJob with an AWS SDK or CLI in the *Amazon S3 API Reference*.

Using S3 Batch Operations to turn off S3 Object Lock legal holds

The following example builds on the previous examples of creating a trust policy, and setting S3 Batch Operations and S3 Object Lock configuration permissions. This example shows how to disable Object Lock legal hold on objects using Batch Operations.

The example first updates the role to grant `s3:PutObjectLegalHold` permissions, creates a Batch Operations job that turns off (removes) legal hold from the objects identified in the manifest, and then reports on it.

To use the following examples, replace the *`user input placeholders`* with your own information.

Using the AWS CLI

The following AWS CLI examples show how to use Batch Operations to turn off S3 Object Lock legal holds across multiple objects.

Example— Updates the role to grant s3:PutObjectLegalHold permissions

```
export AWS_PROFILE='aws-user'

read -d '' legal_hold_permissions <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObjectLegalHold"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
            ]
        }
    ]

EOF

aws iam put-role-policy --role-name batch_operations-objectlock --policy-name legal-
hold-permissions --policy-document "${legal_hold_permissions}"
```

Example— Turn off legal hold

The following example turns off legal hold.

```
export AWS_PROFILE='aws-user'
export AWS_DEFAULT_REGION='us-west-2'
export ACCOUNT_ID=123456789012
```

```
export ROLE_ARN='arn:aws:iam::123456789012:role/batch_operations-objectlock'

read -d '' OPERATION <<EOF
{
  "S3PutObjectLegalHold": {
    "LegalHold": {
      "Status":"OFF"
    }
  }
}
EOF

read -d '' MANIFEST <<EOF
{
  "Spec": {
    "Format": "S3BatchOperations_CSV_20180820",
    "Fields": [
      "Bucket",
      "Key"
    ]
  },
  "Location": {
    "ObjectArn": "arn:aws:s3:::amzn-s3-demo-manifest-bucket/legalhold-object-
manifest.csv",
    "ETag": "Your-manifest-ETag"
  }
}
EOF

read -d '' REPORT <<EOF
{
  "Bucket": "arn:aws:s3:::amzn-s3-demo-completion-report-bucket",
  "Format": "Report_CSV_20180820",
  "Enabled": true,
  "Prefix": "reports/legalhold-objects-batch_operations",
  "ReportScope": "AllTasks"
}
EOF

aws \
    s3control create-job \
    --account-id "${ACCOUNT_ID}" \
    --manifest "${MANIFEST//$'\n'}" \
    --operation "${OPERATION//$'\n'/}" \
```

```
    --report "${REPORT//$'\n'}" \
    --priority 10 \
    --role-arn "${ROLE_ARN}" \
    --client-request-token "$(uuidgen)" \
    --region "${AWS_DEFAULT_REGION}" \
    --description "Turn off legal hold";
```

Using the AWS SDK for Java

For examples of how to use S3 Batch Operations to turn off S3 Object Lock legal hold with the AWS SDK for Java, see Use CreateJob with an AWS SDK or CLI in the *Amazon S3 API Reference*.

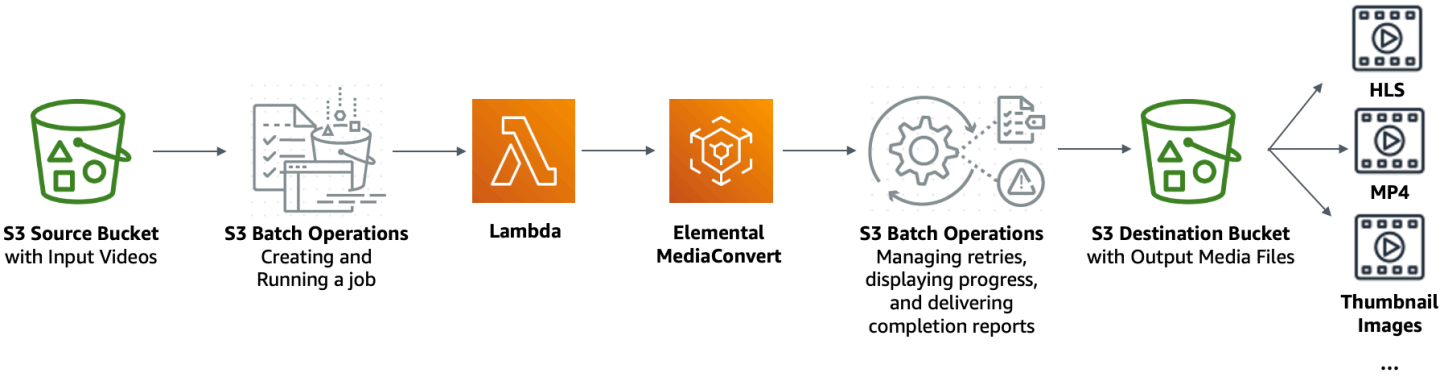
Tutorial: Batch-transcoding videos with S3 Batch Operations

Video consumers use devices of all shapes, sizes, and vintages to enjoy media content. This wide array of devices presents a challenge for content creators and distributors. Instead of being in a one-size-fits-all format, videos must be converted so that they can span a broad range of sizes, formats, and bitrates. This conversion task is even more challenging when you have a large number of videos that must be converted.

AWS offers you a method to build a scalable, distributed architecture that does the following:

- Ingests input videos
- Processes the videos for playback on a wide range of devices
- Stores the transcoded media files
- Delivers the output media files to meet demand

When you have extensive video repositories stored in Amazon S3, you can transcode these videos from their source formats into multiple file types in the size, resolution, and format needed by a particular video player or device. Specifically, S3 Batch Operations provides you with a solution to invoke AWS Lambda functions for existing input videos in an S3 source bucket. Then, the Lambda functions call AWS Elemental MediaConvert to perform large-scale video transcoding tasks. The converted output media files are stored in an S3 destination bucket.



Objective

In this tutorial, you learn how to set up S3 Batch Operations to invoke a Lambda function for batch-transcoding of videos stored in an S3 source bucket. The Lambda function calls MediaConvert to transcode the videos. The outputs for each video in the S3 source bucket are as follows:

- An HTTP Live Streaming (HLS) adaptive bitrate stream for playback on devices of multiple sizes and varying bandwidths
- An MP4 video file
- Thumbnail images collected at intervals

Topics

- Prerequisites
- Step 1: Create an S3 bucket for the output media files
- Step 2: Create an IAM role for MediaConvert
- Step 3: Create an IAM role for your Lambda function
- Step 4: Create a Lambda function for video transcoding
- Step 5: Configure Amazon S3 Inventory for your S3 source bucket
- Step 6: Create an IAM role for S3 Batch Operations
- Step 7: Create and run an S3 Batch Operations job
- Step 8: Check the output media files from your S3 destination bucket
- Step 9: Clean up
- Next steps

Prerequisites

Before you start this tutorial, you must have an Amazon S3 source bucket (for example, *amzn-s3-demo-source-bucket*) with videos to be transcoded already stored in it.

You can give the bucket another name if you want. For more information about bucket names in Amazon S3, see General purpose bucket naming rules.

For the S3 source bucket, keep the settings related to **Block Public Access settings for this bucket** set to the defaults (**Block *all* public access** is enabled). For more information, see Creating a general purpose bucket.

For more information about uploading videos to the S3 source bucket, see Uploading objects. If you're uploading many large video files to S3, you might want to use Amazon S3 Transfer Acceleration to configure fast and secure file transfers. Transfer Acceleration can speed up video uploading to your S3 bucket for long-distance transfer of larger videos. For more information, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

Step 1: Create an S3 bucket for the output media files

In this step, you create an S3 destination bucket to store the converted output media files. You also create a Cross Origin Resource Sharing (CORS) configuration to allow cross-origin access to the transcoded media files stored in your S3 destination bucket.

Substeps

- Create a bucket for the output media files
- Add a CORS configuration to the S3 output bucket

Create a bucket for the output media files

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.
4. For **Bucket name**, enter a name for your bucket (for example, *amzn-s3-demo-destination-bucket1*).
5. For **Region**, choose the AWS Region where you want the bucket to reside.

6. To ensure public access to your output media files, in **Block Public Access settings for this bucket**, clear **Block *all* public access**.

   > **Warning**
   >
   > Before you complete this step, review Blocking public access to your Amazon S3 storage to ensure that you understand and accept the risks involved with allowing public access. When you turn off Block Public Access settings to make your bucket public, anyone on the internet can access your bucket. We recommend that you block all public access to your buckets.
   > If you don't want to clear the Block Public Access settings, you can use Amazon CloudFront to deliver the transcoded media files to viewers (end users). For more information, see Tutorial: Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53.

7. Select the check box next to **I acknowledge that the current settings might result in this bucket and the objects within becoming public.**
8. Keep the remaining settings set to the defaults.
9. Choose **Create bucket**.

Add a CORS configuration to the S3 output bucket

A JSON CORS configuration defines a way for client web applications (video players in this context) that are loaded in one domain to play transcoded output media files in a different domain.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you created earlier (for example, *amzn-s3-demo-destination-bucket1*).
4. Choose the **Permissions** tab.
5. In the **Cross-origin resource sharing (CORS)** section, choose **Edit**.
6. In the CORS configuration text box, copy and paste the following CORS configuration.

   The CORS configuration must be in JSON format. In this example, the `AllowedOrigins` attribute uses the wildcard character (*) to specify all origins. If you know your specific

origin, you can restrict the `AllowedOrigins` attribute to your specific player URL. For more information about configuring this and other attributes, see Elements of a CORS configuration.

```
[
    {
        "AllowedOrigins": [
            "*"
        ],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedHeaders": [
            "*"
        ],
        "ExposeHeaders": []

    }
]
```

7. Choose **Save changes**.

Step 2: Create an IAM role for MediaConvert

To use AWS Elemental MediaConvert to transcode input videos stored in your S3 bucket, you must have an AWS Identity and Access Management (IAM) service role to grant MediaConvert permissions to read and write video files from and to your S3 source and destination buckets. When you run transcoding jobs, the MediaConvert console uses this role.

To create an IAM role for MediaConvert

1. Create an IAM role with a role name that you choose (for example, **`tutorial-mediaconvert-role`**). To create this role, follow the steps in Create your MediaConvert role in IAM (console) in the *AWS Elemental MediaConvert User Guide*.
2. After you create the IAM role for MediaConvert, in the list of **Roles**, choose the name of the role for MediaConvert that you created (for example, **`tutorial-mediaconvert-role`**).
3. On the **Summary** page, copy the **Role ARN** (which starts with `arn:aws:iam::`), and save the ARN for use later.

   For more information about ARNs, see Amazon Resource Names (ARNs) in the *AWS General Reference*.

Step 3: Create an IAM role for your Lambda function

To batch-transcode videos with MediaConvert and S3 Batch Operations, you use a Lambda function to connect these two services to convert videos. This Lambda function must have an IAM role that grants the Lambda function permissions to access MediaConvert and S3 Batch Operations.

Substeps

- Create an IAM role for your Lambda function
- Embed an inline policy for the IAM role of your Lambda function

Create an IAM role for your Lambda function

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**, and then choose **Create role**.
3. Choose the **AWS service** role type, and then under **Common use cases**, choose **Lambda**.
4. Choose **Next: Permissions**.
5. On the **Attach permissions policies** page, enter **`AWSLambdaBasicExecutionRole`** in the **Filter policies** box. To attach the managed policy **AWSLambdaBasicExecutionRole** to this role to grant write permissions to Amazon CloudWatch Logs, select the check box next to **AWSLambdaBasicExecutionRole**.
6. Choose **Next**.
7. For **Role name**, enter **`tutorial-lambda-transcode-role`**.
8. (Optional) Add tags to the managed policy.
9. Choose **Create role**.

Embed an inline policy for the IAM role of your Lambda function

To grant permissions to the MediaConvert resource that's needed for the Lambda function to execute, you must use an inline policy.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**.

3. In the **Roles** list, choose the name of the IAM role that you created earlier for your Lambda function (for example, **`tutorial-lambda-transcode-role`**).
4. Choose the **Permissions** tab.
5. Choose **Add inline policy**.
6. Choose the **JSON** tab, and then copy and paste the following JSON policy.

   In the JSON policy, replace the example ARN value of `Resource` with the role ARN of the IAM role for MediaConvert that you created in Step 2 (for example, **`tutorial-mediaconvert-role`**).

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*",
            "Effect": "Allow",
            "Sid": "Logging"
        },
        {
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/tutorial-mediaconvert-role"
            ],
            "Effect": "Allow",
            "Sid": "PassRole"
        },
        {
            "Action": [
                "mediaconvert:*"
            ],
            "Resource": [
                "*"
```

```
            ],
            "Effect": "Allow",
            "Sid": "MediaConvertService"
        },
        {
            "Action": [
                "s3:*"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "S3Service"
        }
    ]
}
```

7. Choose **Review Policy**.
8. For **Name**, enter **`tutorial-lambda-policy`**.
9. Choose **Create Policy**.

   After you create an inline policy, it is automatically embedded in the IAM role of your Lambda function.

Step 4: Create a Lambda function for video transcoding

In this section of the tutorial, you build a Lambda function using the AWS SDK for Python (Boto3) to integrate with S3 Batch Operations and MediaConvert. To start transcoding the videos already stored in your S3 source bucket, you run an S3 Batch Operations job that directly invokes the Lambda function for each video in the S3 source bucket. Then, the Lambda function submits a transcoding job for each video to MediaConvert.

Substeps

- Write Lambda function code and create a deployment package
- Create a Lambda function with an execution role (console)
- Deploy your Lambda function with .zip file archives and configure the Lambda function (console)

Write Lambda function code and create a deployment package

1. On your local machine, create a folder named `batch-transcode`.
2. In the `batch-transcode` folder, create a file with JSON job settings. For example, you can use the settings provided in this section, and name the file `job.json`.

   A `job.json` file specifies the following:

   - Which files to transcode
   - How you want to transcode your input videos
   - What output media files you want to create
   - What to name the transcoded files
   - Where to save the transcoded files
   - Which advanced features to apply, and so on

   In this tutorial, we use the following `job.json` file to create the following outputs for each video in the S3 source bucket:

   - An HTTP Live Streaming (HLS) adaptive bitrate stream for playback on multiple devices of differing sizes and varying bandwidths
   - An MP4 video file
   - Thumbnail images collected at intervals

   This example `job.json` file uses Quality-Defined Variable Bitrate (QVBR) to optimize video quality. The HLS output is Apple-compliant (audio unmixed from video, segment duration of 6 seconds, and optimized video quality through auto QVBR).

   If you don't want to use the example settings provided here, you can generate a `job.json` specification based on your use case. To ensure consistency across your outputs, make sure that your input files have similar video and audio configurations. For any input files with different video and audio configurations, create separate automations (unique `job.json` settings). For more information, see Example AWS Elemental MediaConvert job settings in JSON in the *AWS Elemental MediaConvert User Guide*.

```
{
  "OutputGroups": [
```

```
    {
      "CustomName": "HLS",
      "Name": "Apple HLS",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
                484,
                485,
                486,
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 640,
            "ScalingBehavior": "DEFAULT",
            "Height": 360,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
```

```
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 1200000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_360"
```

```
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {
              "AudioFramesPerPes": 4,
              "PcrControl": "PCR_EVERY_PES_PACKET",
              "PmtPid": 480,
              "PrivateMetadataPid": 503,
              "ProgramNumber": 1,
              "PatInterval": 0,
              "PmtInterval": 0,
              "TimedMetadata": "NONE",
              "TimedMetadataPid": 502,
              "VideoPid": 481,
              "AudioPids": [
                482,
                483,
                484,
                485,
                486,
                487,
                488,
                489,
                490,
                491,
                492
              ]
            }
          },
          "VideoDescription": {
            "Width": 960,
            "ScalingBehavior": "DEFAULT",
            "Height": 540,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
```

```
                "GopClosedCadence": 1,
                "GopSize": 2,
                "Slices": 1,
                "GopBReference": "DISABLED",
                "MaxBitrate": 3500000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_540"
        },
        {
```

```
            "ContainerSettings": {
              "Container": "M3U8",
              "M3u8Settings": {
                "AudioFramesPerPes": 4,
                "PcrControl": "PCR_EVERY_PES_PACKET",
                "PmtPid": 480,
                "PrivateMetadataPid": 503,
                "ProgramNumber": 1,
                "PatInterval": 0,
                "PmtInterval": 0,
                "TimedMetadata": "NONE",
                "VideoPid": 481,
                "AudioPids": [
                  482,
                  483,
                  484,
                  485,
                  486,
                  487,
                  488,
                  489,
                  490,
                  491,
                  492
                ]
              }
            },
            "VideoDescription": {
              "Width": 1280,
              "ScalingBehavior": "DEFAULT",
              "Height": 720,
              "TimecodeInsertion": "DISABLED",
              "AntiAlias": "ENABLED",
              "Sharpness": 50,
              "CodecSettings": {
                "Codec": "H_264",
                "H264Settings": {
                  "InterlaceMode": "PROGRESSIVE",
                  "NumberReferenceFrames": 3,
                  "Syntax": "DEFAULT",
                  "Softness": 0,
                  "GopClosedCadence": 1,
                  "GopSize": 2,
                  "Slices": 1,
```

```
                "GopBReference": "DISABLED",
                "MaxBitrate": 5000000,
                "SlowPal": "DISABLED",
                "SpatialAdaptiveQuantization": "ENABLED",
                "TemporalAdaptiveQuantization": "ENABLED",
                "FlickerAdaptiveQuantization": "DISABLED",
                "EntropyEncoding": "CABAC",
                "FramerateControl": "INITIALIZE_FROM_SOURCE",
                "RateControlMode": "QVBR",
                "CodecProfile": "MAIN",
                "Telecine": "NONE",
                "MinIInterval": 0,
                "AdaptiveQuantization": "HIGH",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "TRANSITION_DETECTION",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "FramerateConversionAlgorithm": "DUPLICATE_DROP",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "INITIALIZE_FROM_SOURCE",
                "NumberBFramesBetweenReferenceFrames": 2,
                "RepeatPps": "DISABLED"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioRenditionSets": "program_audio",
              "SegmentModifier": "$dt$",
              "IFrameOnlyManifest": "EXCLUDE"
            }
          },
          "NameModifier": "_720"
        },
        {
          "ContainerSettings": {
            "Container": "M3U8",
            "M3u8Settings": {}
```

```
          },
          "AudioDescriptions": [
            {
              "AudioSourceName": "Audio Selector 1",
              "CodecSettings": {
                "Codec": "AAC",
                "AacSettings": {
                  "Bitrate": 96000,
                  "CodingMode": "CODING_MODE_2_0",
                  "SampleRate": 48000
                }
              }
            }
          ],
          "OutputSettings": {
            "HlsSettings": {
              "AudioGroupId": "program_audio",
              "AudioTrackType": "ALTERNATE_AUDIO_AUTO_SELECT_DEFAULT"
            }
          },
          "NameModifier": "_audio"
        }
      ],
      "OutputGroupSettings": {
        "Type": "HLS_GROUP_SETTINGS",
        "HlsGroupSettings": {
          "ManifestDurationFormat": "INTEGER",
          "SegmentLength": 6,
          "TimedMetadataId3Period": 10,
          "CaptionLanguageSetting": "OMIT",
          "Destination": "s3://EXAMPLE-BUCKET/HLS/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          },
          "TimedMetadataId3Frame": "PRIV",
          "CodecSpecification": "RFC_4281",
          "OutputSelection": "MANIFESTS_AND_SEGMENTS",
          "ProgramDateTimePeriod": 600,
          "MinSegmentLength": 0,
          "DirectoryStructure": "SINGLE_DIRECTORY",
```

```
          "ProgramDateTime": "EXCLUDE",
          "SegmentControl": "SEGMENTED_FILES",
          "ManifestCompression": "NONE",
          "ClientCache": "ENABLED",
          "StreamInfResolution": "INCLUDE"
        }
      }
    },
    {
      "CustomName": "MP4",
      "Name": "File Group",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "MP4",
            "Mp4Settings": {
              "CslgAtom": "INCLUDE",
              "FreeSpaceBox": "EXCLUDE",
              "MoovPlacement": "PROGRESSIVE_DOWNLOAD"
            }
          },
          "VideoDescription": {
            "Width": 1280,
            "ScalingBehavior": "DEFAULT",
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 100,
            "CodecSettings": {
              "Codec": "H_264",
              "H264Settings": {
                "InterlaceMode": "PROGRESSIVE",
                "ParNumerator": 1,
                "NumberReferenceFrames": 3,
                "Syntax": "DEFAULT",
                "Softness": 0,
                "GopClosedCadence": 1,
                "HrdBufferInitialFillPercentage": 90,
                "GopSize": 2,
                "Slices": 2,
                "GopBReference": "ENABLED",
                "HrdBufferSize": 10000000,
                "MaxBitrate": 5000000,
                "ParDenominator": 1,
```

```
                "EntropyEncoding": "CABAC",
                "RateControlMode": "QVBR",
                "CodecProfile": "HIGH",
                "MinIInterval": 0,
                "AdaptiveQuantization": "AUTO",
                "CodecLevel": "AUTO",
                "FieldEncoding": "PAFF",
                "SceneChangeDetect": "ENABLED",
                "QualityTuningLevel": "SINGLE_PASS_HQ",
                "UnregisteredSeiTimecode": "DISABLED",
                "GopSizeUnits": "SECONDS",
                "ParControl": "SPECIFIED",
                "NumberBFramesBetweenReferenceFrames": 3,
                "RepeatPps": "DISABLED",
                "DynamicSubGop": "ADAPTIVE"
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          },
          "AudioDescriptions": [
            {
              "AudioTypeControl": "FOLLOW_INPUT",
              "AudioSourceName": "Audio Selector 1",
              "CodecSettings": {
                "Codec": "AAC",
                "AacSettings": {
                  "AudioDescriptionBroadcasterMix": "NORMAL",
                  "Bitrate": 160000,
                  "RateControlMode": "CBR",
                  "CodecProfile": "LC",
                  "CodingMode": "CODING_MODE_2_0",
                  "RawFormat": "NONE",
                  "SampleRate": 48000,
                  "Specification": "MPEG4"
                }
              },
              "LanguageCodeControl": "FOLLOW_INPUT",
              "AudioType": 0
            }
          ]
        }
```

```
      ],
      "OutputGroupSettings": {
        "Type": "FILE_GROUP_SETTINGS",
        "FileGroupSettings": {
          "Destination": "s3://EXAMPLE-BUCKET/MP4/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          }
        }
      }
    },
    {
      "CustomName": "Thumbnails",
      "Name": "File Group",
      "Outputs": [
        {
          "ContainerSettings": {
            "Container": "RAW"
          },
          "VideoDescription": {
            "Width": 1280,
            "ScalingBehavior": "DEFAULT",
            "Height": 720,
            "TimecodeInsertion": "DISABLED",
            "AntiAlias": "ENABLED",
            "Sharpness": 50,
            "CodecSettings": {
              "Codec": "FRAME_CAPTURE",
              "FrameCaptureSettings": {
                "FramerateNumerator": 1,
                "FramerateDenominator": 5,
                "MaxCaptures": 500,
                "Quality": 80
              }
            },
            "AfdSignaling": "NONE",
            "DropFrameTimecode": "ENABLED",
            "RespondToAfd": "NONE",
            "ColorMetadata": "INSERT"
          }
```

```
        }
      ],
      "OutputGroupSettings": {
        "Type": "FILE_GROUP_SETTINGS",
        "FileGroupSettings": {
          "Destination": "s3://EXAMPLE-BUCKET/Thumbnails/",
          "DestinationSettings": {
            "S3Settings": {
              "AccessControl": {
                "CannedAcl": "PUBLIC_READ"
              }
            }
          }
        }
      }
    },
  ],
  "AdAvailOffset": 0,
  "Inputs": [
    {
      "AudioSelectors": {
        "Audio Selector 1": {
          "Offset": 0,
          "DefaultSelection": "DEFAULT",
          "ProgramSelection": 1
        }
      },
      "VideoSelector": {
        "ColorSpace": "FOLLOW"
      },
      "FilterEnable": "AUTO",
      "PsiControl": "USE_PSI",
      "FilterStrength": 0,
      "DeblockFilter": "DISABLED",
      "DenoiseFilter": "DISABLED",
      "TimecodeSource": "EMBEDDED",
      "FileInput": "s3://EXAMPLE-INPUT-BUCKET/input.mp4"
    }
  ]
}
```

3. In the `batch-transcode` folder, create a file with a Lambda function. You can use the following Python example and name the file `convert.py`.

S3 Batch Operations sends specific task data to a Lambda function and requires result data back. For request and response examples for the Lambda function, information about response and result codes, and example Lambda functions for S3 Batch Operations, see Invoke AWS Lambda function.

```
import json
import os
from urllib.parse import urlparse
import uuid
import boto3

"""
When you run an S3 Batch Operations job, your job
invokes this Lambda function. Specifically, the Lambda function is
invoked on each video object listed in the manifest that you specify
for the S3 Batch Operations job in Step 5.

Input parameter "event": The S3 Batch Operations event as a request
                         for the Lambda function.

Input parameter "context": Context about the event.

Output: A result structure that Amazon S3 uses to interpret the result
        of the operation. It is a job response returned back to S3 Batch
 Operations.
"""
def handler(event, context):

    invocation_schema_version = event['invocationSchemaVersion']
    invocation_id = event['invocationId']
    task_id = event['tasks'][0]['taskId']

    source_s3_key = event['tasks'][0]['s3Key']
    source_s3_bucket = event['tasks'][0]['s3BucketArn'].split(':::')[-1]
    source_s3 = 's3://' + source_s3_bucket + '/' + source_s3_key

    result_list = []
    result_code = 'Succeeded'
    result_string = 'The input video object was converted successfully.'

    # The type of output group determines which media players can play
    # the files transcoded by MediaConvert.
```

```
    # For more information, see Creating outputs with AWS Elemental MediaConvert.
    output_group_type_dict = {
        'HLS_GROUP_SETTINGS': 'HlsGroupSettings',
        'FILE_GROUP_SETTINGS': 'FileGroupSettings',
        'CMAF_GROUP_SETTINGS': 'CmafGroupSettings',
        'DASH_ISO_GROUP_SETTINGS': 'DashIsoGroupSettings',
        'MS_SMOOTH_GROUP_SETTINGS': 'MsSmoothGroupSettings'
    }

    try:
        job_name = 'Default'
        with open('job.json') as file:
            job_settings = json.load(file)

        job_settings['Inputs'][0]['FileInput'] = source_s3

        # The path of each output video is constructed based on the values of
        # the attributes in each object of OutputGroups in the job.json file.
        destination_s3 = 's3://{0}/{1}/{2}' \
            .format(os.environ['amzn-s3-demo-destination-bucket'],
                    os.path.splitext(os.path.basename(source_s3_key))[0],
                    os.path.splitext(os.path.basename(job_name))[0])

        for output_group in job_settings['OutputGroups']:
            output_group_type = output_group['OutputGroupSettings']['Type']
            if output_group_type in output_group_type_dict.keys():
                output_group_type = output_group_type_dict[output_group_type]
                output_group['OutputGroupSettings'][output_group_type]
['Destination'] = \
                    "{0}{1}".format(destination_s3,
                                    urlparse(output_group['OutputGroupSettings']
[output_group_type]['Destination']).path)
            else:
                raise ValueError("Exception: Unknown Output Group Type {}."
                                 .format(output_group_type))

        job_metadata_dict = {
            'assetID': str(uuid.uuid4()),
            'application': os.environ['Application'],
            'input': source_s3,
            'settings': job_name
        }

        region = os.environ['AWS_DEFAULT_REGION']
```

```
        endpoints = boto3.client('mediaconvert', region_name=region) \
            .describe_endpoints()
        client = boto3.client('mediaconvert', region_name=region,
                              endpoint_url=endpoints['Endpoints'][0]['Url'],
                              verify=False)

        try:
            client.create_job(Role=os.environ['MediaConvertRole'],
                              UserMetadata=job_metadata_dict,
                              Settings=job_settings)
        # You can customize error handling based on different error codes that
        # MediaConvert can return.
        # For more information, see MediaConvert error codes.
        # When the result_code is TemporaryFailure, S3 Batch Operations retries
        # the task before the job is completed. If this is the final retry,
        # the error message is included in the final report.
        except Exception as error:
            result_code = 'TemporaryFailure'
            raise

    except Exception as error:
        if result_code != 'TemporaryFailure':
            result_code = 'PermanentFailure'
        result_string = str(error)

    finally:
        result_list.append({
            'taskId': task_id,
            'resultCode': result_code,
            'resultString': result_string,
        })

    return {
        'invocationSchemaVersion': invocation_schema_version,
        'treatMissingKeyAs': 'PermanentFailure',
        'invocationId': invocation_id,
        'results': result_list
    }
```

4. To create a deployment package with `convert.py` and `job.json` as a `.zip` file named `lambda.zip`, in your local terminal, open the `batch-transcode` folder that you created earlier, and run the following command.

For **macOS users**, run the following command:

```
zip -r lambda.zip convert.py job.json
```

For **Windows users**, run the following commands:

```
powershell Compress-Archive convert.py lambda.zip
```

```
powershell Compress-Archive -update job.json lambda.zip
```

Create a Lambda function with an execution role (console)

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. In the left navigation pane, choose **Functions**.
3. Choose **Create function**.
4. Choose **Author from scratch**.
5. Under **Basic information**, do the following:
   a. For **Function name**, enter **`tutorial-lambda-convert`**.
   b. For **Runtime**, choose **Python 3.13**.
6. Choose **Change default execution role**, and under **Execution role**, choose **Use an existing role**.
7. Under **Existing role**, choose the name of the IAM role that you created for your Lambda function in Step 3 (for example, **`tutorial-lambda-transcode-role`**).
8. For the remaining settings, keep the defaults.
9. Choose **Create function**.

Deploy your Lambda function with .zip file archives and configure the Lambda function (console)

1. In the **Code Source** section of the page for the Lambda function that you created (for example, **`tutorial-lambda-convert`**), choose **Upload from** and then **.zip file**.
2. Choose **Upload** to select your local `.zip` file.

3. Choose the `lambda.zip` file that you created earlier, and choose **Open**.
4. Choose **Save**.
5. In the **Runtime settings** section, choose **Edit**.
6. To tell the Lambda runtime which handler method in your Lambda function code to invoke, enter **`convert.handler`** in the **Handler** field.

   When you configure a function in Python, the value of the handler setting is the file name and the name of the handler module, separated by a dot (`.`). For example, `convert.handler` calls the `handler` method defined in the `convert.py` file.
7. Choose **Save**.
8. On your Lambda function page, choose the **Configuration** tab. In the left navigation pane on the **Configuration** tab, choose **Environment variables**, and then choose **Edit**.
9. Choose **Add environment variable**. Then, enter the specified **Key** and **Value** for each of the following environment variables:

   - **Key**: **`DestinationBucket`** **Value**: **`amzn-s3-demo-destination-bucket1`**

     This value is the S3 bucket for output media files that you created in Step 1.
   - **Key**: **`MediaConvertRole`** **Value**: **`arn:aws:iam::`*`111122223333`*`:role/tutorial-mediaconvert-role`**

     This value is the ARN of the IAM role for MediaConvert that you created in Step 2. Make sure to replace this ARN with the actual ARN of your IAM role.
   - **Key**: **`Application`** **Value**: **`Batch-Transcoding`**

     This value is the name of the application.
10. Choose **Save**.
11. (Optional) On the **Configuration** tab, in the **General configuration** section of the left navigation pane, choose **Edit**. In the **Timeout** field, enter **2** min **0** sec. Then, choose **Save**.

    Timeout is the amount of time that Lambda allows a function to run for an invocation before stopping it. The default is 3 seconds. Pricing is based on the amount of memory configured and the amount of time that your code runs. For more information, see AWS Lambda pricing.

Step 5: Configure Amazon S3 Inventory for your S3 source bucket

After setting up the transcoding Lambda function, create an S3 Batch Operations job to transcode a set of videos. First, you need a list of input video objects that you want S3 Batch Operations to run the specified transcoding action on. To get a list of input video objects, you can generate an S3 Inventory report for your S3 source bucket (for example, *amzn-s3-demo-source-bucket*).

Substeps

- Create and configure a bucket for S3 Inventory reports for input videos
- Configure Amazon S3 Inventory for your S3 video source bucket
- Check the inventory report for your S3 video source bucket

Create and configure a bucket for S3 Inventory reports for input videos

To store an S3 Inventory report that lists the objects of the S3 source bucket, create an S3 Inventory destination bucket, and then configure a bucket policy for the bucket to write inventory files to the S3 source bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.
4. For **Bucket name**, enter a name for your bucket (for example, *amzn-s3-demo-destination-bucket2*).
5. For **AWS Region**, choose the AWS Region where you want the bucket to reside.

   The inventory destination bucket must be in the same AWS Region as the source bucket where you are setting up S3 Inventory. The inventory destination bucket can be in a different AWS account.
6. In **Block Public Access settings for this bucket**, keep the default settings (**Block *all* public access** is enabled).
7. For the remaining settings, keep the defaults.
8. Choose **Create bucket**.
9. In the **Buckets** list, choose the name of the bucket that you just created (for example, *amzn-s3-demo-destination-bucket2*).

10. To grant Amazon S3 permission to write data for the inventory reports to the S3 Inventory destination bucket, choose the **Permissions** tab.
11. Scroll down to the **Bucket policy** section, and choose **Edit**. The **Bucket policy** page opens.
12. To grant permissions for S3 Inventory, in the **Policy** field, paste the following bucket policy.

    Replace the three example values with the following values:

    - The name of the bucket that you created to store the inventory reports (for example, *amzn-s3-demo-destination-bucket2*).
    - The name of the source bucket that stores the input videos (for example, *amzn-s3-demo-source-bucket*).
    - The AWS account ID that you used to create the S3 video source bucket (for example, *111122223333*.

    JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"InventoryAndAnalyticsExamplePolicy",
      "Effect":"Allow",
      "Principal": {"Service": "s3.amazonaws.com"},
      "Action":"s3:PutObject",
      "Resource":["arn:aws:s3:::amzn-s3-demo-destination-bucket2/*"],
      "Condition": {
          "ArnLike": {
              "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-bucket"
           },
         "StringEquals": {
             "aws:SourceAccount": "111122223333",
             "s3:x-amz-acl": "bucket-owner-full-control"
          }
       }
    }
  ]
}
```

13. Choose **Save changes**.

Configure Amazon S3 Inventory for your S3 video source bucket

To generate a flat file list of video objects and metadata, you must configure S3 Inventory for your S3 video source bucket. These scheduled inventory reports can include all the objects in the bucket or objects grouped by a shared prefix. In this tutorial, the S3 Inventory report includes all the video objects in your S3 source bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. To configure an S3 Inventory report of the input videos in your S3 source bucket, in the **Buckets** list, choose the name of the S3 source bucket (for example, *amzn-s3-demo-source-bucket*).
4. Choose the **Management** tab.
5. Scroll down to the **Inventory configurations** section, and choose **Create inventory configuration**.
6. For **Inventory configuration name**, enter a name (for example, **tutorial-inventory-config**).
7. Under **Inventory scope**, choose **Current version only** for **Object versions** and keep the other **Inventory scope** settings set to the defaults for this tutorial.
8. In the **Report details** section, for **Destination bucket**, choose **This account**.
9. For **Destination**, choose **Browse S3**, and choose the destination bucket that you created earlier to save the inventory reports to (for example, *amzn-s3-demo-destination-bucket2*). Then choose **Choose path**.

   The inventory destination bucket must be in the same AWS Region as the source bucket where you are setting up S3 Inventory. The inventory destination bucket can be in a different AWS account.

   Under the **Destination** bucket field, the **Destination bucket permission** is added to the inventory destination bucket policy, allowing Amazon S3 to place data in the inventory destination bucket. For more information, see Creating a destination bucket policy.
10. For **Frequency**, choose **Daily**.
11. For **Output format**, choose **CSV**.
12. For **Status**, choose **Enable**.

13. In the **Server-side encryption** section, choose **Disable** for this tutorial.

    For more information, see Configuring inventory by using the S3 console and Granting Amazon S3 permission to use your customer managed key for encryption.

14. In the **Additional fields - *optional*** section, select **Size**, **Last modified**, and **Storage class**.
15. Choose **Create**.

For more information, see Configuring inventory by using the S3 console.

Check the inventory report for your S3 video source bucket

When an inventory report is published, the manifest files are sent to the S3 Inventory destination bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the video source bucket (for example, *amzn-s3-demo-source-bucket*).
4. Choose **Management**.
5. To see if your S3 Inventory report is ready so that you can create an S3 Batch Operations job in Step 7, under **Inventory configurations**, check whether the **Create job from manifest** button is enabled.

   > **Note**
   >
   > It can take up to 48 hours to deliver the first inventory report. If the **Create job from manifest** button is disabled, the first inventory report has not been delivered. Wait until the first inventory report is delivered and the **Create job from manifest** button is enabled before you create an S3 Batch Operations job in Step 7.

6. To check an S3 Inventory report (`manifest.json`), in the **Destination** column, choose the name of the inventory destination bucket that you created earlier for storing inventory reports (for example, *amzn-s3-demo-destination-bucket2*).
7. On the **Objects** tab, choose the existing folder with the name of your S3 source bucket (for example, *amzn-s3-demo-source-bucket*). Then choose the name that you entered in

Inventory configuration name when you created the inventory configuration earlier (for example, **`tutorial-inventory-config`**).

You can see a list of folders with the generation dates of the reports as their names.

8. To check the daily S3 Inventory report for a particular date, choose the folder with the corresponding generation date name, and then choose `manifest.json`.
9. To check the details of the inventory report on a specific date, on the **manifest.json** page, choose **Download** or **Open**.

Step 6: Create an IAM role for S3 Batch Operations

To use S3 Batch Operations to do batch-transcoding, you must first create an IAM role to give Amazon S3 permissions to perform S3 Batch Operations.

Substeps

- Create an IAM policy for S3 Batch Operations
- Create an S3 Batch Operations IAM role and attach permissions policies

Create an IAM policy for S3 Batch Operations

You must create an IAM policy that gives S3 Batch Operations permission to read the input manifest, invoke the Lambda function, and write the S3 Batch Operations job completion report.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policies**.
3. Choose **Create policy**.
4. Choose the **JSON** tab.
5. In the **JSON** text field, paste the following JSON policy.

   In the JSON policy, replace the four example values with the following values:

   - The name of the source bucket that stores your input videos (for example, *`amzn-s3-demo-source-bucket`*).
   - The name of the inventory destination bucket that you created in Step 5 to store `manifest.json` files (for example, *`amzn-s3-demo-destination-bucket2`*).

- The name of the bucket that you created in Step 1 to store output media files (for example, *amzn-s3-demo-destination-bucket1*). In this tutorial, we put job completion reports in the destination bucket for output media files.
- The role ARN of the Lambda function that you created in Step 4. To find and copy the role ARN of the Lambda function, do the following:
  - In a new browser tab, open the **Functions** page on the Lambda console at https://console.aws.amazon.com/lambda/home#/functions.
  - In **Functions** list, choose the name of the Lambda function that you created in Step 4 (for example, **`tutorial-lambda-convert`**).
  - Choose **Copy ARN**.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "S3Get",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
                "arn:aws:s3:::amzn-s3-demo-destination-bucket2/*"
            ]
        },
        {
            "Sid": "S3PutJobCompletionReport",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket1/*"
        },
        {
            "Sid": "S3BatchOperationsInvokeLambda",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction"
```

```
            ],
            "Resource": [
                "arn:aws:lambda:us-west-2:111122223333:function:tutorial-
lambda-convert"
            ]
        }
    ]
}
```

6. Choose **Next: Tags**.
7. Choose **Next: Review**.
8. In the **Name** field, enter **`tutorial-s3batch-policy`**.
9. Choose **Create policy**.

Create an S3 Batch Operations IAM role and attach permissions policies

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**, and then choose **Create role**.
3. Choose the **AWS service** role type, and then choose the **S3** service.
4. Under **Select your use case**, choose **S3 Batch Operations**.
5. Choose **Next**.
6. Under **Attach permissions**, enter the name of the IAM policy that you created earlier (for example, **`tutorial-s3batch-policy`**) in the search box to filter the list of policies. Select the check box next to the name of the policy (for example, **`tutorial-s3batch-policy`**).
7. Choose **Next**.
8. For **Role name**, enter **`tutorial-s3batch-role`**.
9. Choose **Create role**.

   After you create the IAM role for S3 Batch Operations, the following trust policy is automatically attached to the role. This trust policy allows the S3 Batch Operations service principal to assume the IAM role.

   JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

Step 7: Create and run an S3 Batch Operations job

To create an S3 Batch Operations job to process the input videos in your S3 source bucket, you must specify parameters for this particular job.

> **Note**
>
> Before you start creating an S3 Batch Operations job, make sure that the **Create job from manifest** button is enabled. For more information, see Check the inventory report for your S3 video source bucket. If the **Create job from manifest** button is disabled, the first inventory report has not been delivered and you must wait until the button is enabled. After you configure Amazon S3 Inventory for your S3 source bucket in Step 5, it can take up to 48 hours to deliver the first inventory report.

Substeps

- Create an S3 Batch Operations job
- Run the S3 Batch Operations job to invoke your Lambda function
- (Optional) Check your completion report
- (Optional) Monitor each Lambda invocation in the Lambda console
- (Optional) Monitor each MediaConvert video-transcoding job in the MediaConvert console

Create an S3 Batch Operations job

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. Choose **Create job**.
4. For **AWS Region**, choose the Region where you want to create your job.

   In this tutorial, to use the S3 Batch Operations job to invoke a Lambda function, you must create the job in the same Region as the S3 video source bucket where the objects referenced in the manifest are located.
5. In the **Manifest** section, do the following:

   a. For **Manifest format**, choose **S3 Inventory report (manifest.json)**.

   b. For **Manifest object**, choose **Browse S3** to find the bucket that you created in Step 5 for storing inventory reports (for example, *amzn-s3-demo-destination-bucket2*). On the **Manifest object** page, navigate through the object names until you find a `manifest.json` file for a specific date. This file lists the information about all the videos that you want to batch-transcode. When you've found the `manifest.json` file that you want to use, choose the option button next to it. Then choose **Choose path**.

   c. (Optional) For **Manifest object version ID - *optional***, enter the version ID for the manifest object if you want to use a version other than the most recent.
6. Choose **Next**.
7. To use the Lambda function to transcode all the objects listed in the selected `manifest.json` file, under **Operation type**, choose **Invoke AWS Lambda function**.
8. In the **Invoke Lambda function** section, do the following:

   a. Choose **Choose from functions in your account**.

   b. For **Lambda function**, choose the Lambda function that you created in Step 4 (for example, **`tutorial-lambda-convert`**).

   c. For **Lambda function version**, keep the default value **$LATEST**.
9. Choose **Next**. The **Configure additional options** page opens.
10. In the **Additional options** section, keep the default settings.

    For more information about these options, see Batch Operations job request elements.

11. In the **Completion report** section, for **Path to completion report destination**, choose **Browse S3**. Find the bucket that you created for output media files in Step 1 (for example, *amzn-s3-demo-destination-bucket1*). Choose the option button next to that bucket's name. Then choose **Choose path**.

    For the remaining **Completion report** settings, keep the defaults. For more information about completion report settings, see Batch Operations job request elements. A completion report maintains a record of the job's details and the operations performed.

12. In the **Permissions** section, choose **Choose from existing IAM roles**. For **IAM role**, choose the IAM role for your S3 Batch Operations job that you created in Step 6 (for example, **`tutorial-s3batch-role`**).

13. Choose **Next**.

14. On the **Review** page, review the settings. Then choose **Create job**.

    After S3 finishes reading your S3 Batch Operations job's manifest, it sets the **Status** of the job to **Awaiting your confirmation to run**. To see updates to the job's status, refresh the page. You can't run your job until its status is **Awaiting your confirmation to run**.

Run the S3 Batch Operations job to invoke your Lambda function

Run your Batch Operations job to invoke your Lambda function for video transcoding. If your job fails, you can check your completion report to identify the cause.

To run the S3 Batch Operations job

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. In the **Jobs** list, choose the **Job ID** of the job on the first row, which is the S3 Batch Operations job that you created earlier.
4. Choose **Run job**.
5. Review your job parameters again, and confirm that the value for **Total objects listed in manifest** is the same as the number of objects in the manifest. Then choose **Run job**.

   Your S3 Batch Operations job page opens.

6. After the job starts running, on your job page, under **Status**, check the progress of your S3 Batch Operations job, such as **Status**, **% Complete**, **Total succeeded (rate)**, **Total failed (rate)**, **Date terminated**, and **Reason for termination**.

   When the S3 Batch Operations job completes, view the data on your job page to confirm that the job finished as expected.

   If more than 50 percent of an S3 Batch Operations job's object operations fail after more than 1,000 operations have been attempted, the job automatically fails. To check your completion report to identify the cause of the failures, use the following optional procedure.

(Optional) Check your completion report

You can use your completion report to determine which objects failed and the cause of the failures.

To check your completion report for details about failed objects

1. On the page of your S3 Batch Operations job, scroll down to the **Completion report** section, and choose the link under **Completion report destination**.

   The S3 output destination bucket's page opens.
2. On the **Objects** tab, choose the folder that has a name ending with the job ID of the S3 Batch Operations job that you created earlier.
3. Choose **results/**.
4. Select the check box next to the `.csv` file.
5. To view the job report, choose **Open** or **Download**.

(Optional) Monitor each Lambda invocation in the Lambda console

After the S3 Batch Operations job starts running, the job invokes the Lambda function for each input video object. S3 writes logs of each Lambda invocation to CloudWatch Logs. You can use the Lambda console's monitoring dashboard to monitor your Lambda function.

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. In the left navigation pane, choose **Functions**.
3. In the **Functions** list, choose the name of the Lambda function that you created in Step 4 (for example, **`tutorial-lambda-convert`**).

4. Choose the **Monitor** tab.
5. Under **Metrics**, see the runtime metrics for your Lambda function.
6. Under **Logs**, view log data for each Lambda invocation through CloudWatch Logs Insights.

> **Note**
>
> When you use S3 Batch Operations with a Lambda function, the Lambda function is invoked on each object. If your S3 Batch Operations job is large, it can invoke multiple Lambda functions at the same time, causing a spike in Lambda concurrency.
> Each AWS account has a Lambda concurrency quota per Region. For more information, see AWS Lambda Function Scaling in the *AWS Lambda Developer Guide*. A best practice for using Lambda functions with S3 Batch Operations is to set a concurrency limit on the Lambda function itself. Setting a concurrency limit keeps your job from consuming most of your Lambda concurrency and potentially throttling other functions in your account. For more information, see Managing Lambda reserved concurrency in the *AWS Lambda Developer Guide*.

(Optional) Monitor each MediaConvert video-transcoding job in the MediaConvert console

A MediaConvert job does the work of transcoding a media file. When your S3 Batch Operations job invokes your Lambda function for each video, each Lambda function invocation creates a MediaConvert transcoding job for each input video.

1. Sign in to the AWS Management Console and open the MediaConvert console at https://console.aws.amazon.com/mediaconvert/.
2. If the MediaConvert introductory page appears, choose **Get started**.
3. From the list of **Jobs**, view each row to monitor the transcoding task for each input video.
4. Identify the row of a job that you want to check, and choose the **Job ID** link to open the job details page.
5. On the **Job summary** page, under **Outputs**, choose the link for the HLS, MP4, or Thumbnails output, depending on what is supported by your browser, to go to the S3 destination bucket for the output media files.
6. In the corresponding folder (HLS, MP4, or Thumbnails) of your S3 output destination bucket, choose the name of the output media file object.

   The object's detail page opens.

7. On the object's detail page, under **Object overview**, choose the link under **Object URL** to watch the transcoded output media file.

Step 8: Check the output media files from your S3 destination bucket

To check the output media files from your S3 destination bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the S3 destination bucket for output media files that you created in Step 1 (for example, *amzn-s3-demo-destination-bucket1*).
4. On the **Objects** tab, each input video has a folder that has the name of the input video. Each folder contains the transcoded output media files for an input video.

   To check the output media files for an input video, do the following:

   a. Choose the folder with the name of the input video that you want to check.
   b. Choose the **Default/** folder.
   c. Choose the folder for a transcoded format (HLS, MP4, or thumbnails in this tutorial).
   d. Choose the name of the output media file.
   e. To watch the transcoded file, on the object's details page, choose the link under **Object URL**.

      Output media files in the HLS format are split into short segments. To play these videos, embed the object URL of the `.m3u8` file in a compatible player.

Step 9: Clean up

If you transcoded videos using S3 Batch Operations, Lambda, and MediaConvert only as a learning exercise, delete the AWS resources that you allocated so that you no longer accrue charges.

Substeps

- Delete the S3 Inventory configuration for your S3 source bucket
- Delete the Lambda function
- Delete the CloudWatch log group

- Delete the IAM roles together with the inline policies for the IAM roles
- Delete the customer-managed IAM policy
- Empty the S3 buckets
- Delete the S3 buckets

Delete the S3 Inventory configuration for your S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of your source bucket (for example, *amzn-s3-demo-source-bucket*).
4. Choose the **Management** tab.
5. In the **Inventory configurations** section, choose the option button next to the inventory configuration that you created in Step 5 (for example, **tutorial-inventory-config**).
6. Choose **Delete**, and then choose **Confirm**.

Delete the Lambda function

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. In the left navigation pane, choose **Functions**.
3. Select the check box next to the function that you created in Step 4 (for example, **tutorial-lambda-convert**).
4. Choose **Actions**, and then choose **Delete**.
5. In the **Delete function** dialog box, choose **Delete**.

Delete the CloudWatch log group

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Logs**, and then choose **Log groups**.
3. Select the check box next to the log group that has a name ending with the Lambda function that you created in Step 4 (for example, **tutorial-lambda-convert**).

4. Choose **Actions**, and then choose **Delete log group(s)**.
5. In the **Delete log group(s)** dialog box, choose **Delete**.

Delete the IAM roles together with the inline policies for the IAM roles

To delete the IAM roles that you created in Step 2, Step 3, and Step 6, do the following:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Roles**, and then select the check boxes next to the role names that you want to delete.
3. Choose **Delete**.
4. In the confirmation dialog box, enter the required response in the text input field based on the prompt, and choose **Delete**.

Delete the customer-managed IAM policy

To delete the customer-managed IAM policy that you created in Step 6, do the following:

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policies**.
3. Choose the option button next to the policy that you created in Step 6 (for example, **`tutorial-s3batch-policy`**). You can use the search box to filter the list of policies.
4. Choose **Actions**, and then choose **Delete**.
5. Confirm that you want to delete this policy by entering its name in the text field, and then choose **Delete**.

Empty the S3 buckets

To empty the S3 buckets that you created in Prerequisites, Step 1, and Step 5, do the following:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.

3. In the **Buckets** list, choose the option button next to the name of the bucket that you want to empty, and then choose **Empty**.
4. On the **Empty bucket** page, confirm that you want to empty the bucket by entering **`permanently delete`** in the text field, and then choose **Empty**.

Delete the S3 buckets

To delete the S3 buckets that you created in Prerequisites, Step 1, and Step 5, do the following:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the option button next to the name of the bucket that you want to delete.
4. Choose **Delete**.
5. On the **Delete bucket** page, confirm that you want to delete the bucket by entering the bucket name in the text field, and then choose **Delete bucket**.

Next steps

After completing this tutorial, you can further explore other relevant use cases:

- You can use Amazon CloudFront to stream the transcoded media files to viewers across the globe. For more information, see Tutorial: Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53.
- You can transcode videos at the moment when you upload them to the S3 source bucket. To do so, you can configure an Amazon S3 event trigger that automatically invokes the Lambda function to transcode new objects in S3 with MediaConvert. For more information, see Tutorial: Using an Amazon S3 trigger to invoke a Lambda function in the *AWS Lambda Developer Guide*.

Troubleshooting S3 Batch Operations

Amazon S3 Batch Operations allows you to perform large-scale operations on Amazon S3 objects. This guide helps you troubleshoot common issues you might encounter.

To troubleshoot issues with S3 Batch Replication, see Troubleshooting replication.

There are two primary types of failures that result in Batch operation errors:

1. **API Failure** – The requested API (such as `CreateJob`) has failed to execute.
2. **Job Failure** – The initial API request succeeded but the job failed, for example, due to issues with the manifest or permissions to objects specified in the manifest.

NoSuchJobException

Type: API failure

The `NoSuchJobException` occurs when S3 Batch Operations cannot locate the specified job. This error can happen in several scenarios beyond simple job expiration. Common causes include the following.

1. **Job expiration** – Jobs are automatically deleted 90 days after reaching a terminal state (`Complete`, `Cancelled`, or `Failed`).
2. **Incorrect job ID** – The job ID used in `DescribeJob` or `UpdateJobStatus` doesn't match the ID returned by `CreateJob`.
3. **Wrong region** – Attempting to access a job in a different region than where it was created.
4. **Wrong account** – Using a job ID from a different AWS account.
5. **Job ID format errors** – Typos, extra characters, or incorrect formatting in the job ID.
6. **Timing issues** – Checking job status immediately after creation before the job is fully registered.

Related error messages include the following.

1. `No such job`
2. `The specified job does not exist`

Best practices to prevent `NoSuchJobException` API failures

1. **Store job IDs immediately** – Save the job ID from the `CreateJob` response before making subsequent API calls.
2. **Implement retry logic** – Add exponential backoff when checking job status immediately after creation.
3. **Set up monitoring** – Create CloudWatch alarms to track job completion before the 90-day expiration. For details, see Using CloudWatch alarms in the Amazon CloudWatch User Guide.

4. **Use consistent regions** – Ensure all job operations use the same region as job creation.
5. **Validate input** – Check job ID format before making API calls.

When jobs expire

Jobs in terminal states are automatically deleted after 90 days. To avoid losing job information, consider the following.

1. **Download completion reports before expiration** – For instructions on retrieving and storing job results, see Completion reports.
2. **Archive job metadata in your own systems** – Store critical job information in your databases or monitoring systems.
3. **Set up automated notifications before the 90-day deadline** – Use Amazon EventBridge to create rules that trigger notifications when jobs complete. For more information, see Amazon S3 Event Notifications.

`NoSuchJobException` troubleshooting

1. Use the following command to verify the job exists in your account and region.

```
aws s3control list-jobs --account-id 111122223333 --region us-east-1
```

2. Use the following command to search across all job statuses. Possible job statuses include `Active`, `Cancelled`, `Cancelling`, `Complete`, `Completing`, `Failed`, `Failing`, `New`, `Paused`, `Pausing`, `Preparing`, `Ready`, and `Suspended`.

```
aws s3control list-jobs --account-id 111122223333 --job-statuses your-job-status
```

3. Use the following command to check if the job exists in other regions where you commonly create jobs.

```
aws s3control list-jobs --account-id 111122223333 --region job-region-1 aws
 s3control list-jobs --account-id 111122223333 --region job-region-2
```

4. Validate the job ID format. Job IDs typically contain 36 character, such as `12345678-1234-1234-1234-123456789012`. Check for extra spaces, missing characters, or case sensitivity issues and verify you're using the complete job ID returned by the `CreateJob` command.
5. Use the following command to check CloudTrail logs for job creation events.

```
    aws logs filter-log-events --log-group-name CloudTrail/S3BatchOperations \ --
filter-pattern "{ $.eventName = CreateJob }" \ --start-time timestamp
```

AccessDeniedException

Type: API failure

The `AccessDeniedException` occurs when an S3 Batch Operations request is blocked due to insufficient permissions, unsupported operations, or policy restrictions. This is one of the most common errors in Batch Operations. It has the following common causes:

1. **Missing IAM permissions** – The IAM identity lacks required permissions for Batch Operations APIs.
2. **Insufficient S3 permissions** – Missing permissions to access source or destination buckets and objects.
3. **Job execution role issues** – The job execution role lacks permissions to perform the specified operation.
4. **Unsupported operations** – Attempting to use operations not supported in the current region or bucket type.
5. **Cross-account access issues** – Missing permissions for cross-account bucket or object access.
6. **Resource-based policy restrictions** – Bucket policies or object ACLs blocking the operation.
7. **Service Control Policy (SCP) restrictions** – Organization-level policies preventing the operation.

Related error messages:

1. `Access Denied`

2. `User: arn:aws:iam::account:user/username is not authorized to perform: s3:operation`
3. `Cross-account pass role is not allowed`
4. `The bucket policy does not allow the specified operation`

Best practices to prevent AccessDeniedException API failures

1. **Use least privilege principle** – Grant only the minimum permissions required for your specific operations.
2. **Test permissions before large jobs** – Run small test jobs to validate permissions before processing thousands of objects.
3. **Use IAM policy simulator** – Test policies before deployment using the IAM policy simulator. For more information, see IAM policy testing with the IAM policy simulator in the IAM User Guide.
4. **Implement proper cross-account setup** – Check your cross-account access configuration for cross-account job configurations. For more information, see IAM tutorial: Delegate access across AWS accounts using IAM roles in the IAM User Guide.
5. **Monitor permission changes** – Set up CloudTrail alerts for IAM policy modifications that might affect Batch Operations.
6. **Document role requirements** – Maintain clear documentation of required permissions for each job type.
7. **Use common permission templates** - Use the permission examples and policy templates:
   a. Granting permissions for Batch Operations
   b. Cross account resources in IAM in the IAM User Guide.
   c. Control access to VPC endpoints using endpoint policies in the AWS PrivateLink Guide.

AccessDeniedException troubleshooting

Follow these steps systematically to identify and resolve permission issues.

1. Check Operations supported by S3 Batch Operations for supported operations by region. Confirm directory bucket operations are only available at Regional and Zonal endpoints. Verify the operation is supported for your bucket's storage class.
2. Use the following command to determine if you can list jobs.

```
aws s3control list-jobs --account-id 111122223333
```

3. Use the following command to check IAM permissions for the requesting identity. The account running the job needs the following permissions: `s3:CreateJob`, `s3:DescribeJob`, `s3:ListJobs`, `s3:UpdateJobPriority`, `s3:UpdateJobStatus`, and `iam:PassRole`.

```
aws sts get-caller-identity 111122223333
```

4. Use the following command to check if the role exists and is assumable.

```
aws iam get-role --role-name role-name
```

5. Use the following command to review the role's trust policy. The role running the job must have the following:

   1. A trust relationship allowing `batchoperations.s3.amazonaws.com` to assume the role.
   2. The operations the batch operation is performing (such as `s3:PutObjectTagging` for tagging operations).
   3. Access to the source and destination buckets.
   4. Permission to read the manifest file.
   5. Permission to write completion reports.

```
aws iam get-role --role-name role-name --query 'Role.AssumeRolePolicyDocument'
```

6. Use the following command to test access to the manifest and source buckets.

```
aws s3 ls s3://amzn-s3-demo-bucket
```

7. Test the operation being performed by the batch operation. For example, if the batch operation performs tagging, tag a sample object in the source bucket.

8. Review bucket policies for policies that might deny the operation.
    1. Check object ACLs if working with legacy access controls.
    2. Verify no Service Control Policies (SCPs) are blocking the operation.
    3. Confirm VPC endpoint policies allow Batch Operations if using VPC endpoints.
9. Use the following command to use CloudTrail to identify permission failures.

```
aws logs filter-log-events --log-group-name CloudTrail/S3BatchOperations \
    --filter-pattern "{ $.errorCode = AccessDenied }" \
    --start-time timestamp
```

SlowDownError

Type: API failure

The `SlowDownError` exception occurs when your account has exceeded the request rate limit for S3 Batch Operations APIs. This is a throttling mechanism to protect the service from being overwhelmed by too many requests. It has the following common causes:

1. **High API request frequency** – Making too many API calls in a short time period.
2. **Concurrent job operations** – Multiple applications or users creating/managing jobs simultaneously.
3. **Automated scripts without rate limiting** – Scripts that don't implement proper backoff strategies.
4. **Polling job status too frequently** – Checking job status more often than necessary.
5. **Burst traffic patterns** – Sudden spikes in API usage during peak processing times.
6. **Regional capacity limits** – Exceeding the allocated request capacity for your region.

Related error messages:

1. `SlowDown`
2. `Please reduce your request rate`
3. `Request rate exceeded`

Best practices to prevent SlowDownError API failures

1. **Implement client-side rate limiting** – Add delays between API calls in your applications.
2. **Use exponential backoff with jitter** – Randomize retry delays to avoid thundering herd problems.
3. **Set up proper retry logic** – Implement automatic retries with increasing delays for transient errors.
4. **Use event-driven architectures** – Replace polling with EventBridge notifications for job status changes.
5. **Distribute load across time** – Stagger job creation and status checks across different time periods.
6. **Monitor and alert on rate limits** – Set up CloudWatch alarms to detect when you're approaching limits.

Most AWS SDKs include built-in retry logic for rate limiting errors. Configure them as follows:

1. **AWS CLI** – Use `cli-read-timeout` and `cli-connect-timeout` parameters.
2. **AWS SDK for Python (Boto3)** – Configure retry modes and max attempts in your client configuration.
3. **AWS SDK for Java** – Use `RetryPolicy` and `ClientConfiguration` settings.
4. **AWS SDK for JavaScript** – Configure `maxRetries` and `retryDelayOptions`.

For more information about retry patterns and best practices, see Retry with backoff pattern in the AWS Prescriptive Guidance guide.

SlowDownError troubleshooting

1. In your code, implement exponential backoff immediately.

   Example of exponential backoff in bash

```
for attempt in {1..5}; do
    if aws s3control describe-job --account-id 111122223333 --job-id job-id; then
        break
    else
```

```
        wait_time=$((2**attempt)) echo "Rate limited, waiting ${wait_time}
 seconds..." sleep $wait_time
        fi
done
```

2. Use CloudTrail to identify the source of high request volume.

```
aws logs filter-log-events \
    --log-group-name CloudTrail/S3BatchOperations \
    --filter-pattern "{ $.eventName = CreateJob || $.eventName = DescribeJob }" \
    --start-time timestamp \
    --query 'events[*].[eventTime,sourceIPAddress,userIdentity.type,eventName]'
```

3. Review polling frequency.

    1. Avoid checking job status more than once every 30 seconds for active jobs.
    2. Use job completion notifications instead of polling when possible.
    3. Implement jitter in your polling intervals to avoid synchronized requests.

4. Optimize your API usage patterns.

    1. Batch multiple operations when possible.
    2. Use `ListJobs` to get status of multiple jobs in one call.
    3. Cache job information to reduce redundant API calls.
    4. Spread job creation across time rather than creating many jobs simultaneously.

5. Use CloudWatch metrics for API calls to monitor your request patterns.

```
   aws logs put-metric-filter \
       --log-group-name CloudTrail/S3BatchOperations \
       --filter-name S3BatchOpsAPICallCount \
       --filter-pattern "{ $.eventSource = s3.amazonaws.com && $.eventName =
 CreateJob }" \
       --metric-transformations \
       metricName=S3BatchOpsAPICalls,metricNamespace=Custom/
S3BatchOps,metricValue=1
```

InvalidManifestContent

Type: Job failure

The `InvalidManifestContent` exception occurs when there are issues with the manifest file format, content, or structure that prevent S3 Batch Operations from processing the job. It has the following common causes:

1. **Format violations** – Missing required columns, incorrect delimiters, or malformed CSV structure.
2. **Content encoding issues** – Incorrect character encoding, BOM markers, or non-UTF-8 characters.
3. **Object key problems** – Invalid characters, improper URL encoding, or keys exceeding length limits.
4. **Size limitations** – Manifest contains more objects than the operation supports.
5. **Version ID format errors** – Malformed or invalid version IDs for versioned objects.
6. **ETag format issues** – Incorrect ETag format or missing quotes for operations that require ETags.
7. **Inconsistent data** – Mixed formats within the same manifest or inconsistent column counts.

Related error messages:

1. `Required fields are missing in the schema: + missingFields`
2. `Invalid Manifest Content`
3. `The S3 Batch Operations job failed because it contains more keys than the maximum allowed in a single job`
4. `Invalid object key format`
5. `Manifest file is not properly formatted`
6. `Invalid version ID format`
7. `ETag format is invalid`

Best practices to prevent InvalidManifestContent job failures

1. **Validate before upload** – Test manifest format with small jobs before processing large datasets.
2. **Use consistent encoding** – Always use UTF-8 encoding without BOM for manifest files.
3. **Implement manifest generation standards** – Create templates and validation procedures for manifest creation.

4. **Handle special characters properly** – URL encode object keys that contain special characters.
5. **Monitor object counts** – Track manifest size and split large jobs proactively.
6. **Validate object existence** – Verify objects exist before including them in manifests.
7. **Use AWS tools for manifest generation** – Leverage AWS CLI `s3api list-objects-v2` to generate properly formatted object lists.

Common manifest issues and solutions:

1. **Missing required columns** – Ensure your manifest includes all required columns for your operation type. The most common missing columns are Bucket and Key.
2. **Incorrect CSV format** – Use comma delimiters, ensure consistent column counts across all rows, and avoid embedded line breaks within fields.
3. **Special characters in object keys** – URL encode object keys that contain spaces, Unicode characters, or XML special characters (<, >, &, ", ').
4. **Large manifest files** – Split manifests with more than the operation limit into multiple smaller manifests and create separate jobs.
5. **Invalid version IDs** – Ensure version IDs are properly formatted alphanumeric strings. Remove version ID column if not needed.
6. **Encoding issues** – Save manifest files as UTF-8 without BOM. Avoid copying manifests through systems that might alter encoding.

For detailed manifest format specifications and examples, see the following:

1. Specifying a manifest
2. Operations supported by S3 Batch Operations
3. Naming Amazon S3 objects

InvalidManifestContent troubleshooting

1. Download and inspect the manifest file. Manually verify the manifest meets format requirements:
    1. CSV format with comma delimiters.
    2. UTF-8 encoding without BOM.

3. Consistent number of columns across all rows.
4. No empty lines or trailing spaces.
5. Object keys properly URL encoded if they contain special characters.

Use the following command to download the manifest file.

```
aws s3 cp s3://amzn-s3-demo-bucket1/manifest-key ./manifest.csv
```

2. Check required columns for your operation:
    1. All operations: `Bucket`, `Key`
    2. Copy operations: `VersionId` (optional)
    3. Restore operations: `VersionId` (optional)
    4. Replace tag operations: No additional columns required.
    5. Replace ACL operations: No additional columns required.
    6. Initiate restore: `VersionId` (optional)
3. Check object count limits:
    1. Copy: 1 billion objects maximum.
    2. Delete: 1 billion objects maximum.
    3. Restore: 1 billion objects maximum.
    4. Tagging: 1 billion objects maximum.
    5. ACL: 1 billion objects maximum.
4. Create a test manifest with a few objects from your original manifest.
5. Use the following command to check if a sample of objects from the manifest exist.

```
aws s3 ls s3://amzn-s3-demo-bucket1/object-key
```

6. Check job failure details and review the failure reason and any specific error details in the job description.

```
aws s3control describe-job --account-id 111122223333 --job-id job-id
```

Querying data in place with Amazon S3 Select

Important

Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

With Amazon S3 Select, you can use structured query language (SQL) statements to filter the contents of an Amazon S3 object and retrieve only the subset of data that you need. By using Amazon S3 Select to filter this data, you can reduce the amount of data that Amazon S3 transfers, which reduces the cost and latency to retrieve this data.

Amazon S3 Select only allows you to query one object at a time. It works on an object stored in CSV, JSON, or Apache Parquet format. It also works with an object that is compressed with GZIP or BZIP2 (for CSV and JSON objects only), and a server-side encrypted object. You can specify the format of the results as either CSV or JSON, and you can determine how the records in the result are delimited.

You pass SQL expressions to Amazon S3 in the request. Amazon S3 Select supports a subset of SQL. For more information about the SQL elements that are supported by Amazon S3 Select, see SQL reference for Amazon S3 Select.

You can perform SQL queries by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the `SelectObjectContent` REST API operation, or the AWS SDKs.

Note

The Amazon S3 console limits the amount of data returned to 40 MB. To retrieve more data, use the AWS CLI or the API.

Requirements and limits

The following are requirements for using Amazon S3 Select:

- You must have `s3:GetObject` permission for the object you are querying.
- If the object you are querying is encrypted with server-side encryption with customer-provided keys (SSE-C), you must use `https`, and you must provide the encryption key in the request.

The following limits apply when using Amazon S3 Select:

- S3 Select can query only one object per request.
- S3 Select supports querying files up to 5 TB in size.
- The maximum length of a SQL expression is 256 KB.
- The maximum length of a record in the input or result is 1 MB.
- Amazon S3 Select can only emit nested data by using the JSON output format.
- You cannot query an object stored in the S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, or Reduced Redundancy Storage (RRS) storage classes. You also cannot query an object stored in the S3 Intelligent-Tiering Archive Access tier or the S3 Intelligent-Tiering Deep Archive Access tier. For more information about storage classes, see Understanding and managing Amazon S3 storage classes.

Additional limitations apply when using Amazon S3 Select with a Parquet object:

- Amazon S3 Select supports only columnar compression using GZIP or Snappy. Amazon S3 Select doesn't support whole-object compression for a Parquet object.
- Amazon S3 Select doesn't support Parquet output. You must specify the output format as CSV or JSON.
- The maximum uncompressed row group size is 512 MB.
- You must use the data types that are specified in the object's schema.
- Selecting on a repeated field returns only the last value.

Constructing a request

When you construct a request, you provide details of the object that is being queried by using an `InputSerialization` object. You provide details of how the results are to be returned by using

an `OutputSerialization` object. You also include the SQL expression that Amazon S3 uses to filter the request.

For more information about constructing an Amazon S3 Select request, see SelectObjectContent in the *Amazon Simple Storage Service API Reference*. You can also see one of the SDK code examples in the following sections.

Requests using scan ranges

With Amazon S3 Select, you can scan a subset of an object by specifying a range of bytes to query. This capability lets you parallelize scanning the whole object by splitting the work into separate Amazon S3 Select requests for a series of non-overlapping scan ranges.

Scan ranges don't need to be aligned with record boundaries. An Amazon S3 Select scan range request runs across the byte range that you specify. A record that starts within the specified scan range but extends beyond that scan range will be processed by the query. For example, the following shows an Amazon S3 object that contains a series of records in a line-delimited CSV format:

```
A,B
C,D
D,E
E,F
G,H
I,J
```

Suppose that you're using the Amazon S3 Select `ScanRange` parameter and *Start* at (Byte) 1 and *End* at (Byte) 4. So the scan range would start at "`,`" and scan until the end of the record starting at `C`. Your scan range request will return the result `C, D` because that is the end of the record.

Amazon S3 Select scan range requests support Parquet, CSV (without quoted delimiters), or JSON objects (in `LINES` mode only). CSV and JSON objects must be uncompressed. For line-based CSV and JSON objects, when a scan range is specified as part of the Amazon S3 Select request, all records that start within the scan range are processed. For Parquet objects, all of the row groups that start within the scan range requested are processed.

Amazon S3 Select scan range requests are available to use with the AWS CLI, Amazon S3 API, and AWS SDKs. You can use the `ScanRange` parameter in the Amazon S3 Select request for this feature. For more information, see SelectObjectContent in the *Amazon Simple Storage Service API Reference*.

Errors

Amazon S3 Select returns an error code and associated error message when an issue is encountered while attempting to run a query. For a list of error codes and descriptions, see the List of SELECT Object Content Error Codes section of the *Error Responses* page in the *Amazon Simple Storage Service API Reference*.

For more information about Amazon S3 Select, see the following topics.

Topics

- Examples of using Amazon S3 Select on an object
- SQL reference for Amazon S3 Select

Examples of using Amazon S3 Select on an object

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

You can use S3 Select to select content from one object by using the Amazon S3 console, the REST API, and the AWS SDKs.

For more information about supported SQL functions for S3 Select, see SQL functions.

Using the S3 console

To select content from an object in the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the bucket that contains the object that you want to select content from, and then choose the name of the object.
4. Choose **Object actions**, and choose **Query with S3 Select**.
5. Configure **Input settings**, based on the format of your input data.

6. Configure **Output settings**, based on the format of the output that you want to receive.
7. To extract records from the chosen object, under **SQL query**, enter the SELECT SQL commands. For more information on how to write SQL commands, see SQL reference for Amazon S3 Select.
8. After you enter SQL queries, choose **Run SQL query**. Then, under **Query results**, you can see the results of your SQL queries.

Using the REST API

You can use the AWS SDKs to select content from an object. However, if your application requires it, you can send REST requests directly. For more information about the request and response format, see SelectObjectContent.

Using the AWS SDKs

You can use Amazon S3 Select to select some of the content of an object by using the `selectObjectContent` method. If this method is successful, it returns the results of the SQL expression.

Java

To use Amazon S3 Select with the AWS SDK for Java, you can return the value of the first column for each record that is stored in an object that contains data stored in CSV format. You can also request `Progress` and `Stats` messages to be returned. You must provide a valid bucket name and an object that contains data in CSV format.

To use Amazon S3 Select with the AWS SDK for Java, you can return the value of the first column for each record that is stored in an object that contains data stored in CSV format. You can also request `Progress` and `Stats` messages to be returned. You must provide a valid bucket name and an object that contains data in CSV format.

For examples of how to use Amazon S3 Select with the AWS SDK for Java, see Select content from an object in the *Amazon S3 API Reference*.

JavaScript

For a JavaScript example that uses the AWS SDK for JavaScript with the S3 `SelectObjectContent` API operation to select records from JSON and CSV files that are stored in Amazon S3, see the blog post Introducing support for Amazon S3 Select in the AWS SDK for JavaScript.

Python

For a Python example of using SQL queries to search through data that was loaded to Amazon S3 as a comma-separated value (CSV) file by using S3 Select, see the blog post Querying data without servers or databases using Amazon S3 Select.

SQL reference for Amazon S3 Select

Important

Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

This reference contains a description of the structured query language (SQL) elements that are supported by Amazon S3 Select.

Topics

- SELECT command
- Data types
- Operators
- Reserved keywords
- SQL functions

SELECT command

Important

Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports only the SELECT SQL command. The following ANSI standard clauses are supported for SELECT:

- SELECT list

- FROM clause
- WHERE clause
- LIMIT clause

> **Note**
>
> Amazon S3 Select queries currently do not support subqueries or joins.

SELECT list

The SELECT list names the columns, functions, and expressions that you want the query to return. The list represents the output of the query.

```
SELECT *
SELECT projection1 AS column_alias_1, projection2 AS column_alias_2
```

The first form of SELECT with the * (asterisk) returns every row that passed the WHERE clause, as-is. The second form of SELECT creates a row with user-defined output scalar expressions ***projection1*** and ***projection2*** for each column.

FROM clause

Amazon S3 Select supports the following forms of the FROM clause:

```
FROM table_name
FROM table_name alias
FROM table_name AS alias
```

In each form of the FROM clause, `table_name` is the `S3Object` that's being queried. Users coming from traditional relational databases can think of this as a database schema that contains multiple views over a table.

Following standard SQL, the FROM clause creates rows that are filtered in the WHERE clause and projected in the SELECT list.

For JSON objects that are stored in Amazon S3 Select, you can also use the following forms of the FROM clause:

```
FROM S3Object[*].path
FROM S3Object[*].path alias
FROM S3Object[*].path AS alias
```

Using this form of the FROM clause, you can select from arrays or objects within a JSON object. You can specify path by using one of the following forms:

- By name (in an object): .*name* or ['*name*']
- By index (in an array): [*index*]
- By wildcard character (in an object): .*
- By wildcard character (in an array): [*]

Note

- This form of the FROM clause works only with JSON objects.
- Wildcard characters always emit at least one record. If no record matches, then Amazon S3 Select emits the value MISSING. During output serialization (after the query finishes running), Amazon S3 Select replaces MISSING values with empty records.
- Aggregate functions (AVG, COUNT, MAX, MIN, and SUM) skip MISSING values.
- If you don't provide an alias when using a wildcard character, you can refer to the row by using the last element in the path. For example, you could select all prices from a list of books by using the query SELECT price FROM S3Object[*].books[*].price. If the path ends in a wildcard character instead of a name, then you can use the value _1 to refer to the row. For example, instead of SELECT price FROM S3Object[*].books[*].price, you could use the query SELECT _1.price FROM S3Object[*].books[*].
- Amazon S3 Select always treats a JSON document as an array of root-level values. Thus, even if the JSON object that you are querying has only one root element, the FROM clause must begin with S3Object[*]. However, for compatibility reasons, Amazon S3 Select allows you to omit the wildcard character if you don't include a path. Thus, the complete clause FROM S3Object is equivalent to FROM S3Object[*] as S3Object. If you include a path, you must also use the wildcard character. So, FROM S3Object and FROM S3Object[*].*path* are both valid clauses, but FROM S3Object.*path* is not.

Example

Examples:

Example #1

This example shows results when using the following dataset and query:

```
{ "Rules": [ {"id": "1"}, {"expr": "y > x"}, {"id": "2", "expr": "z = DEBUG"} ]}
{ "created": "June 27", "modified": "July 6" }
```

```
SELECT id FROM S3Object[*].Rules[*].id
```

```
{"id":"1"}
{}
{"id":"2"}
{}
```

Amazon S3 Select produces each result for the following reasons:

- `{"id":"id-1"}` – `S3Object[0].Rules[0].id` produced a match.
- `{}` – `S3Object[0].Rules[1].id` did not match a record, so Amazon S3 Select emitted `MISSING`, which was then changed to an empty record during output serialization and returned.
- `{"id":"id-2"}` – `S3Object[0].Rules[2].id` produced a match.
- `{}` – `S3Object[1]` did not match on `Rules`, so Amazon S3 Select emitted `MISSING`, which was then changed to an empty record during output serialization and returned.

If you don't want Amazon S3 Select to return empty records when it doesn't find a match, you can test for the value `MISSING`. The following query returns the same results as the previous query, but with the empty values omitted:

```
SELECT id FROM S3Object[*].Rules[*].id WHERE id IS NOT MISSING
```

```
{"id":"1"}
{"id":"2"}
```

Example #2

This example shows results when using the following dataset and queries:

```
{ "created": "936864000", "dir_name": "important_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": ".aws" }, { "name": "downloads" } ], "owner": "Amazon
 S3" }
{ "created": "936864000", "dir_name": "other_docs", "files": [ { "name": "." },
 { "name": ".." }, { "name": "my stuff" }, { "name": "backup" } ], "owner": "User" }
```

```
SELECT d.dir_name, d.files FROM S3Object[*] d
```

```
{"dir_name":"important_docs","files":[{"name":"."},{"name":".."},{"name":".aws"},
{"name":"downloads"}]}
{"dir_name":"other_docs","files":[{"name":"."},{"name":".."},{"name":"my stuff"},
{"name":"backup"}]}
```

```
SELECT _1.dir_name, _1.owner FROM S3Object[*]
```

```
{"dir_name":"important_docs","owner":"Amazon S3"}
{"dir_name":"other_docs","owner":"User"}
```

WHERE clause

The WHERE clause follows this syntax:

```
WHERE condition
```

The WHERE clause filters rows based on the *condition*. A condition is an expression that has a Boolean result. Only rows for which the condition evaluates to TRUE are returned in the result.

LIMIT clause

The LIMIT clause follows this syntax:

```
LIMIT number
```

The LIMIT clause limits the number of records that you want the query to return based on *number*.

Attribute access

The SELECT and WHERE clauses can refer to record data by using one of the methods in the following sections, depending on whether the file that is being queried is in CSV or JSON format.

CSV

- **Column Numbers** – You can refer to the *Nth* column of a row with the column name *_N*, where *N* is the column position. The position count starts at 1. For example, the first column is named _1 and the second column is named _2.

  You can refer to a column as *_N* or *alias*.*_N*. For example, _2 and myAlias._2 are both valid ways to refer to a column in the SELECT list and WHERE clause.
- **Column Headers** – For objects in CSV format that have a header row, the headers are available to the SELECT list and WHERE clause. In particular, as in traditional SQL, within SELECT and WHERE clause expressions, you can refer to the columns by *alias*.*column_name* or *column_name*.

JSON

- **Document** – You can access JSON document fields as *alias*.*name*. You can also access nested fields, for example, *alias*.*name1*.*name2*.*name3*.
- **List** – You can access elements in a JSON list by using zero-based indexes with the [] operator. For example, you can access the second element of a list as *alias*[1]. You can combine accessing list elements with fields, for example, *alias*.*name1*.*name2*[1].*name3*.
- **Examples:** Consider this JSON object as a sample dataset:

```
{"name": "Susan Smith",
"org": "engineering",
"projects":
    [
     {"project_name":"project1", "completed":false},
     {"project_name":"project2", "completed":true}
    ]
}
```

Example #1

The following query returns these results:

```
Select s.name from S3Object s
```

```
{"name":"Susan Smith"}
```

Example #2

The following query returns these results:

```
Select s.projects[0].project_name from S3Object s
```

```
{"project_name":"project1"}
```

Case sensitivity of header and attribute names

With Amazon S3 Select, you can use double quotation marks to indicate that column headers (for CSV objects) and attributes (for JSON objects) are case sensitive. Without double quotation marks, object headers and attributes are case insensitive. An error is thrown in cases of ambiguity.

The following examples are either 1) Amazon S3 objects in CSV format with the specified column headers, and with `FileHeaderInfo` set to `"Use"` for the query request; or 2) Amazon S3 objects in JSON format with the specified attributes.

Example #1: The object being queried has the header or attribute `NAME`.

- The following expression successfully returns values from the object. Because there are no quotation marks, the query is case insensitive.

  ```
  SELECT s.name from S3Object s
  ```

- The following expression results in a 400 error `MissingHeaderName`. Because there are quotation marks, the query is case sensitive.

  ```
  SELECT s."name" from S3Object s
  ```

Example #2: The Amazon S3 object being queried has one header or attribute with `NAME` and another header or attribute with `name`.

- The following expression results in a 400 error `AmbiguousFieldName`. Because there are no quotation marks, the query is case insensitive, but there are two matches, so the error is thrown.

```
SELECT s.name from S3Object s
```

- The following expression successfully returns values from the object. Because there are quotation marks, the query is case sensitive, so there is no ambiguity.

```
SELECT s."NAME" from S3Object s
```

Using reserved keywords as user-defined terms

Amazon S3 Select has a set of reserved keywords that are needed to run the SQL expressions used to query object content. Reserved keywords include function names, data types, operators, and so on. In some cases, user-defined terms, such as the column headers (for CSV files) or attributes (for JSON objects), might clash with a reserved keyword. When this happens, you must use double quotation marks to indicate that you are intentionally using a user-defined term that clashes with a reserved keyword. Otherwise a 400 parse error will result.

For the full list of reserved keywords, see Reserved keywords.

The following example is either 1) an Amazon S3 object in CSV format with the specified column headers, with `FileHeaderInfo` set to `"Use"` for the query request, or 2) an Amazon S3 object in JSON format with the specified attributes.

Example: The object being queried has a header or attribute named `CAST`, which is a reserved keyword.

- The following expression successfully returns values from the object. Because quotation marks are used in the query, S3 Select uses the user-defined header or attribute.

```
SELECT s."CAST" from S3Object s
```

- The following expression results in a 400 parse error. Because no quotation marks are used in the query, `CAST` clashes with a reserved keyword.

```
SELECT s.CAST from S3Object s
```

Scalar expressions

Within the WHERE clause and the SELECT list, you can have SQL *scalar expressions*, which are expressions that return scalar values. They have the following form:

- ***literal***

  An SQL literal.
- ***column_reference***

  A reference to a column in the form *column_name* or *alias*.*column_name*.
- ***unary_op expression***

  In this case, ***unary_op*** is an SQL unary operator.
- ***expression binary_op expression***

  In this case, ***binary_op*** is an SQL binary operator.
- ***func_name***

  In this case, ***func_name*** is the name of the scalar function to invoke.
- ***expression*** [NOT] BETWEEN ***expression*** AND ***expression***
- ***expression*** LIKE ***expression*** [ESCAPE ***expression***]

Data types

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports several primitive data types.

Data type conversions

The general rule is to follow the CAST function if it's defined. If CAST is not defined, then all input data is treated as a string. In that case, you must cast your input data into the relevant data types when necessary.

For more information about the CAST function, see CAST.

Supported data types

Amazon S3 Select supports the following set of primitive data types.

Name	Description	Examples
`bool`	A Boolean value, either TRUE or FALSE.	`FALSE`
`int`, `integer`	An 8-byte signed integer in the range -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807.	`100000`
`string`	A UTF8-encoded variable-length string. The default limit is 1 character. The maximum character limit is 2,147,483,647.	`'xyz'`
`float`	An 8-byte floating point number.	`CAST(0.456 AS FLOAT)`
`decimal`, `numeric`	A base-10 number, with a maximum precision of 38 (that is, the maximum number of significant digits), and with a scale within the range of -2^{31} to $2^{31}-1$ (that is, the base-10 exponent). **Note** Amazon S3 Select ignores scale and precision when you provide both at the same time.	`123.456`
`timestamp`	Timestamps represent a specific moment in time, always include a local offset, and are capable of arbitrary precision. In the text format, timestamps follow the W3C note on date and time formats, but they must end with the literal T if the timestamps are not at least whole-day precision. Fractional seconds are allowed, with at least one digit of precision, and an unlimited maximum. Local-time offsets c	`CAST('2007-04-05T14:30Z' AS TIMESTAMP)`

Name	Description	Examples
	an be represented as either hour:minute offsets from UTC, or as the literal Z to denote a local time of UTC. Local-time offsets are required on timestamps with time and are not allowed on date values.	

Supported Parquet types

Amazon S3 Select supports the following Parquet types.

- `DATE`
- `DECIMAL`
- `ENUM`
- `INT(8)`
- `INT(16)`
- `INT(32)`
- `INT(64)`
- `LIST`

> **Note**
>
> For `LIST` Parquet type output, Amazon S3 Select supports only JSON format. However, if the query limits the data to simple values, the `LIST` Parquet type can also be queried in CSV format.

- `STRING`
- `TIMESTAMP` supported precision (`MILLIS`/`MICROS`/`NANOS`)

> **Note**
>
> Timestamps saved as an `INT(96)` are unsupported.
> Because of the range of the `INT(64)` type, timestamps that are using the `NANOS` unit can represent only values between `1677-09-21 00:12:43` and `2262-04-11 23:47:16`. Values outside of this range cannot be represented with the `NANOS` unit.

Mapping of Parquet types to supported data types in Amazon S3 Select

Parquet types	Supported data types
DATE	timestamp
DECIMAL	decimal, numeric
ENUM	string
INT(8)	int, integer
INT(16)	int, integer
INT(32)	int, integer
INT(64)	decimal, numeric
LIST	Each Parquet type in list is mapped to the corresponding data type.
STRING	string
TIMESTAMP	timestamp

Operators

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following operators.

Logical operators

- AND
- NOT
- OR

Comparison operators

- <
- >
- <=
- >=
- =
- <>
- !=
- BETWEEN
- IN – For example: `IN ('a', 'b', 'c')`

Pattern-matching operators

- LIKE
- _ (Matches any character)
- % (Matches any sequence of characters)

Unitary operators

- IS NULL
- IS NOT NULL

Math operators

Addition, subtraction, multiplication, division, and modulo are supported, as follows:

- +

- -
- *
- /
- %

Operator precedence

The following table shows the operators' precedence in decreasing order.

Operator or element	Associativity	Required
-	right	unary minus
*, /, %	left	multiplication, division, modulo
+, -	left	addition, subtraction
IN		set membership
BETWEEN		range containment
LIKE		string pattern matching
<>		less than, greater than
=	right	equality, assignment
NOT	right	logical negation

Operator or element	Associativity	Required
AND	left	logical conjunction
OR	left	logical disjunction

Reserved keywords

Important

Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

The following is the list of reserved keywords for Amazon S3 Select. These keywords include function names, data types, operators, and so on, that are needed to run the SQL expressions that are used to query object content.

```
absolute
action
add
all
allocate
alter
and
any
are
as
asc
assertion
at
authorization
avg
bag
begin
between
bit
```

```
bit_length
blob
bool
boolean
both
by
cascade
cascaded
case
cast
catalog
char
char_length
character
character_length
check
clob
close
coalesce
collate
collation
column
commit
connect
connection
constraint
constraints
continue
convert
corresponding
count
create
cross
current
current_date
current_time
current_timestamp
current_user
cursor
date
day
deallocate
dec
decimal
```

```
declare
default
deferrable
deferred
delete
desc
describe
descriptor
diagnostics
disconnect
distinct
domain
double
drop
else
end
end-exec
escape
except
exception
exec
execute
exists
external
extract
false
fetch
first
float
for
foreign
found
from
full
get
global
go
goto
grant
group
having
hour
identity
immediate
```

```
in
indicator
initially
inner
input
insensitive
insert
int
integer
intersect
interval
into
is
isolation
join
key
language
last
leading
left
level
like
limit
list
local
lower
match
max
min
minute
missing
module
month
names
national
natural
nchar
next
no
not
null
nullif
numeric
octet_length
```

```
of
on
only
open
option
or
order
outer
output
overlaps
pad
partial
pivot
position
precision
prepare
preserve
primary
prior
privileges
procedure
public
read
real
references
relative
restrict
revoke
right
rollback
rows
schema
scroll
second
section
select
session
session_user
set
sexp
size
smallint
some
space
```

```
sql
sqlcode
sqlerror
sqlstate
string
struct
substring
sum
symbol
system_user
table
temporary
then
time
timestamp
timezone_hour
timezone_minute
to
trailing
transaction
translate
translation
trim
true
tuple
union
unique
unknown
unpivot
update
upper
usage
user
using
value
values
varchar
varying
view
when
whenever
where
with
work
```

```
write
year
zone
```

SQL functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following SQL functions.

Topics

- Aggregate functions
- Conditional functions
- Conversion functions
- Date functions
- String functions

Aggregate functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following aggregate functions.

Function	Argument type	Return type
`AVG(`*`expression`*`)`	`INT`, `FLOAT`, `DECIMAL`	`DECIMAL` for an `INT` argument, `FLOAT` for a

Function	Argument type	Return type
		floating-point argument; otherwise the same as the argument data type.
COUNT	-	INT
MAX(*expression*)	INT, DECIMAL	Same as the argument type.
MIN(*expression*)	INT, DECIMAL	Same as the argument type.
SUM(*expression*)	INT, FLOAT, DOUBLE, DECIMAL	INT for an INT argument, FLOAT for a floating-point argument; otherwise, the same as the argument data type.

SUM example

To aggregate the total object sizes of a folder in an S3 Inventory report, use a SUM expression.

The following S3 Inventory report is a CSV file that's compressed with GZIP. There are three columns.

- The first column is the name of the S3 bucket (*DOC-EXAMPLE-BUCKET*) that the S3 Inventory report is for.

- The second column is the object key name that uniquely identifies the object in the bucket.

  The *example-folder/* value in the first row is for the folder *example-folder*. In Amazon S3, when you create a folder in your bucket, S3 creates a 0-byte object with a key that's set to the folder name that you provided.

  The *example-folder/object1* value in the second row is for the object *object1* in the folder *example-folder*.

  The *example-folder/object2* value in the third row is for the object *object2* in the folder *example-folder*.

  For more information about S3 folders, see Organizing objects in the Amazon S3 console by using folders.
- The third column is the object size in bytes.

```
"DOC-EXAMPLE-BUCKET","example-folder/","0"
"DOC-EXAMPLE-BUCKET","example-folder/object1","2011267"
"DOC-EXAMPLE-BUCKET","example-folder/object2","1570024"
```

To use a SUM expression to calculate the total size of the folder *example-folder*, run the SQL query with Amazon S3 Select.

```
SELECT SUM(CAST(_3 as INT)) FROM s3object s WHERE _2 LIKE 'example-folder/%' AND _2 !=
 'example-folder/';
```

Query Result:

```
3581291
```

Conditional functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following conditional functions.

Topics

- CASE
- COALESCE
- NULLIF

CASE

The CASE expression is a conditional expression, similar to `if/then/else` statements found in other languages. CASE is used to specify a result when there are multiple conditions. There are two types of CASE expressions: simple and searched.

In simple CASE expressions, an expression is compared with a value. When a match is found, the specified action in the THEN clause is applied. If no match is found, the action in the ELSE clause is applied.

In searched CASE expressions, each CASE is evaluated based on a Boolean expression, and the CASE statement returns the first matching CASE. If no matching CASE is found among the WHEN clauses, the action in the ELSE clause is returned.

Syntax

> **Note**
>
> Currently, Amazon S3 Select doesn't support ORDER BY or queries that contain new lines. Make sure that you use queries with no line breaks.

The following is a simple CASE statement that's used to match conditions:

```
CASE expression WHEN value THEN result [WHEN...] [ELSE result] END
```

The following is a searched CASE statement that's used to evaluate each condition:

```
CASE WHEN boolean condition THEN result [WHEN ...] [ELSE result] END
```

Examples

> **Note**
>
> If you use the Amazon S3 console to run the following examples and your CSV file contains a header row, choose **Exclude the first line of CSV data**.

Example 1: Use a simple CASE expression to replace `New York City` with `Big Apple` in a query. Replace all other city names with `other`.

```
SELECT venuecity, CASE venuecity WHEN 'New York City' THEN 'Big Apple' ELSE 'other' END
 FROM S3Object;
```

Query result:

```
venuecity        |   case
-----------------+-----------
Los Angeles      | other
New York City    | Big Apple
San Francisco    | other
Baltimore        | other
...
```

Example 2: Use a searched CASE expression to assign group numbers based on the `pricepaid` value for individual ticket sales:

```
SELECT pricepaid, CASE WHEN CAST(pricepaid as FLOAT) < 10000 THEN 'group 1' WHEN
 CAST(pricepaid as FLOAT) > 10000 THEN 'group 2' ELSE 'group 3' END FROM S3Object;
```

Query result:

```
pricepaid |  case
-----------+---------
12624.00 | group 2
10000.00 | group 3
10000.00 | group 3
9996.00 | group 1
9988.00 | group 1
...
```

COALESCE

COALESCE evaluates the arguments in order and returns the first non-unknown value, that is, the first non-null or non-missing value. This function does not propagate null and missing values.

Syntax

```
COALESCE ( expression, expression, ... )
```

Parameters

expression

The target expression that the function operates on.

Examples

```
COALESCE(1)                -- 1
COALESCE(null)             -- null
COALESCE(null, null)       -- null
COALESCE(missing)          -- null
COALESCE(missing, missing) -- null
COALESCE(1, null)          -- 1
COALESCE(null, null, 1)    -- 1
COALESCE(null, 'string')   -- 'string'
COALESCE(missing, 1)       -- 1
```

NULLIF

Given two expressions, NULLIF returns NULL if the two expressions evaluate to the same value; otherwise, NULLIF returns the result of evaluating the first expression.

Syntax

```
NULLIF ( expression1, expression2 )
```

Parameters

expression1, *expression2*

The target expressions that the function operates on.

Examples

```
NULLIF(1, 1)             -- null
NULLIF(1, 2)             -- 1
NULLIF(1.0, 1)           -- null
NULLIF(1, '1')           -- 1
NULLIF([1], [1])         -- null
NULLIF(1, NULL)          -- 1
NULLIF(NULL, 1)          -- null
NULLIF(null, null)       -- null
NULLIF(missing, null)    -- null
NULLIF(missing, missing) -- null
```

Conversion functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following conversion function.

Topics

- CAST

CAST

The CAST function converts an entity, such as an expression that evaluates to a single value, from one type to another.

Syntax

```
CAST ( expression AS data_type )
```

Parameters

expression

A combination of one or more values, operators, and SQL functions that evaluate to a value.

data_type

The target data type, such as INT, to cast the expression to. For a list of supported data types, see Data types.

Examples

```
CAST('2007-04-05T14:30Z' AS TIMESTAMP)
CAST(0.456 AS FLOAT)
```

Date functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following date functions.

Topics

- DATE_ADD
- DATE_DIFF
- EXTRACT
- TO_STRING
- TO_TIMESTAMP
- UTCNOW

DATE_ADD

Given a date part, a quantity, and a timestamp, DATE_ADD returns an updated timestamp by altering the date part by the quantity.

Syntax

```
DATE_ADD( date_part, quantity, timestamp )
```

Parameters

date_part

Specifies which part of the date to modify. This can be one of the following:

- year
- month
- day
- hour
- minute
- second

quantity

The value to apply to the updated timestamp. Positive values for *quantity* add to the timestamp's date_part, and negative values subtract.

timestamp

The target timestamp that the function operates on.

Examples

```
DATE_ADD(year, 5, `2010-01-01T`)                -- 2015-01-01 (equivalent to
 2015-01-01T)
DATE_ADD(month, 1, `2010T`)                     -- 2010-02T (result will add precision
 as necessary)
DATE_ADD(month, 13, `2010T`)                    -- 2011-02T
DATE_ADD(day, -1, `2017-01-10T`)                -- 2017-01-09 (equivalent to
 2017-01-09T)
DATE_ADD(hour, 1, `2017T`)                      -- 2017-01-01T01:00-00:00
DATE_ADD(hour, 1, `2017-01-02T03:04Z`)          -- 2017-01-02T04:04Z
DATE_ADD(minute, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:05:05.006Z
DATE_ADD(second, 1, `2017-01-02T03:04:05.006Z`) -- 2017-01-02T03:04:06.006Z
```

DATE_DIFF

Given a date part and two valid timestamps, DATE_DIFF returns the difference in date parts. The return value is a negative integer when the *date_part* value of *timestamp1* is greater than the *date_part* value of *timestamp2*. The return value is a positive integer when the *date_part* value of *timestamp1* is less than the *date_part* value of *timestamp2*.

Syntax

```
DATE_DIFF( date_part, timestamp1, timestamp2 )
```

Parameters

date_part

Specifies which part of the timestamps to compare. For the definition of date_part, see DATE_ADD.

timestamp1

The first timestamp to compare.

timestamp2

The second timestamp to compare.

Examples

```
DATE_DIFF(year, `2010-01-01T`, `2011-01-01T`)            -- 1
DATE_DIFF(year, `2010T`, `2010-05T`)                     -- 4 (2010T is equivalent to
 2010-01-01T00:00:00.000Z)
DATE_DIFF(month, `2010T`, `2011T`)                       -- 12
DATE_DIFF(month, `2011T`, `2010T`)                       -- -12
DATE_DIFF(day, `2010-01-01T23:00`, `2010-01-02T01:00`) -- 0 (need to be at least 24h
 apart to be 1 day apart)
```

EXTRACT

Given a date part and a timestamp, EXTRACT returns the timestamp's date part value.

Syntax

```
EXTRACT( date_part FROM timestamp )
```

Parameters

date_part

Specifies which part of the timestamps to extract. This can be one of the following:

- YEAR
- MONTH
- DAY
- HOUR
- MINUTE
- SECOND
- TIMEZONE_HOUR
- TIMEZONE_MINUTE

timestamp

The target timestamp that the function operates on.

Examples

```
EXTRACT(YEAR FROM `2010-01-01T`)                                -- 2010
EXTRACT(MONTH FROM `2010T`)                                     -- 1 (equivalent to
 2010-01-01T00:00:00.000Z)
EXTRACT(MONTH FROM `2010-10T`)                                  -- 10
EXTRACT(HOUR FROM `2017-01-02T03:04:05+07:08`)                  -- 3
EXTRACT(MINUTE FROM `2017-01-02T03:04:05+07:08`)                -- 4
EXTRACT(TIMEZONE_HOUR FROM `2017-01-02T03:04:05+07:08`)         -- 7
EXTRACT(TIMEZONE_MINUTE FROM `2017-01-02T03:04:05+07:08`)       -- 8
```

TO_STRING

Given a timestamp and a format pattern, TO_STRING returns a string representation of the timestamp in the given format.

Syntax

```
TO_STRING ( timestamp time_format_pattern )
```

Parameters

timestamp

The target timestamp that the function operates on.

time_format_pattern

A string that has the following special character interpretations:

Format	Example	Description
yy	69	2-digit year
y	1969	4-digit year
yyyy	1969	Zero-padded 4-digit year
M	1	Month of year
MM	01	Zero-padded month of yea r
MMM	Jan	Abbreviated month year n ame
MMMM	January	Full month of year name
MMMMM	J	Month of year first letter (NOTE: This format is not valid for use with the TO_TIMEST AMP function.)
d	2	Day of month (1-31)

Format	Example	Description
dd	02	Zero-padded day of month (01-31)
a	AM	AM or PM of day
h	3	Hour of day (1-12)
hh	03	Zero-padded hour of day (01-12)
H	3	Hour of day (0-23)
HH	03	Zero-padded hour of day (00-23)
m	4	Minute of hour (0-59)
mm	04	Zero-padded minute of hour (00-59)
s	5	Second of minute (0-59)
ss	05	Zero-padded second of minute (00-59)

Format	Example	Description
S	0	Fraction of a second (precision: 0.1, range: 0.0-0.9)
SS	6	Fraction of a second (precision: 0.01, range: 0.0-0.99)
SSS	60	Fraction of a second (precision: 0.001, range: 0.0-0.999)
...	...	...
SSSSSSSSS	60000000	Fraction of a second (maximum precision: 1 nanosecond, range: 0.0-0.999999999)
n	60000000	Nano of a second
X	+07 or Z	Offset in hours, or Z if the offset is 0

Format	Example	Description
XX or XXXX	+0700 or Z	Offset in hours and minutes, or Z if the offset is 0
XXX or XXXXX	+07:00 or Z	Offset in hours and minutes, or Z if the offset is 0
x	7	Offset in hours
xx or xxxx	700	Offset in hours and mi nutes
xxx or xxxxx	+07:00	Offset in hours and mi nutes

Examples

```
TO_STRING(`1969-07-20T20:18Z`,  'MMMM d, y')                    -- "July 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMM d, yyyy')                   -- "Jul 20, 1969"
TO_STRING(`1969-07-20T20:18Z`, 'M-d-yy')                        -- "7-20-69"
TO_STRING(`1969-07-20T20:18Z`, 'MM-d-y')                        -- "07-20-1969"
TO_STRING(`1969-07-20T20:18Z`, 'MMMM d, y h:m a')               -- "July 20, 1969 8:18
 PM"
TO_STRING(`1969-07-20T20:18Z`, 'y-MM-dd''T''H:m:ssX')           --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00Z`, 'y-MM-dd''T''H:m:ssX')     --
 "1969-07-20T20:18:00Z"
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXX')   --
 "1969-07-20T20:18:00+0800"
```

```
TO_STRING(`1969-07-20T20:18+08:00`, 'y-MM-dd''T''H:m:ssXXXXX')  --
 "1969-07-20T20:18:00+08:00"
```

TO_TIMESTAMP

Given a string, TO_TIMESTAMP converts it to a timestamp. TO_TIMESTAMP is the inverse operation of TO_STRING.

Syntax

```
TO_TIMESTAMP ( string )
```

Parameters

string

The target string that the function operates on.

Examples

```
TO_TIMESTAMP('2007T')                          -- `2007T`
TO_TIMESTAMP('2007-02-23T12:14:33.079-08:00') -- `2007-02-23T12:14:33.079-08:00`
```

UTCNOW

UTCNOW returns the current time in UTC as a timestamp.

Syntax

```
UTCNOW()
```

Parameters

UTCNOW takes no parameters.

Examples

```
UTCNOW() -- 2017-10-13T16:02:11.123Z
```

String functions

> **Important**
>
> Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more

Amazon S3 Select supports the following string functions.

Topics

- CHAR_LENGTH, CHARACTER_LENGTH
- LOWER
- SUBSTRING
- TRIM
- UPPER

CHAR_LENGTH, CHARACTER_LENGTH

CHAR_LENGTH (or CHARACTER_LENGTH) counts the number of characters in the specified string.

> **Note**
>
> CHAR_LENGTH and CHARACTER_LENGTH are synonyms.

Syntax

```
CHAR_LENGTH ( string )
```

Parameters

string

The target string that the function operates on.

Examples

```
CHAR_LENGTH('')          -- 0
CHAR_LENGTH('abcdefg')   -- 7
```

LOWER

Given a string, LOWER converts all uppercase characters to lowercase characters. Any non-uppercased characters remain unchanged.

Syntax

```
LOWER ( string )
```

Parameters

string

The target string that the function operates on.

Examples

```
LOWER('AbCdEfG!@#$') -- 'abcdefg!@#$'
```

SUBSTRING

Given a string, a start index, and optionally a length, SUBSTRING returns the substring from the start index up to the end of the string, or up to the length provided.

> **Note**
>
> The first character of the input string has an index position of 1.
>
> - If `start` is < 1, with no length specified, then the index position is set to 1.
> - If `start` is < 1, with a length specified, then the index position is set to `start + length -1`.
> - If `start + length -1` < 0, then an empty string is returned.
> - If `start + length -1` > = 0, then the substring starting at index position 1 with the length `start + length - 1` is returned.

Syntax

```
SUBSTRING( string FROM start [ FOR length ] )
```

Parameters

string

The target string that the function operates on.

start

The start position of the string.

length

The length of the substring to return. If not present, proceed to the end of the string.

Examples

```
SUBSTRING("123456789", 0)      -- "123456789"
SUBSTRING("123456789", 1)      -- "123456789"
SUBSTRING("123456789", 2)      -- "23456789"
SUBSTRING("123456789", -4)     -- "123456789"
SUBSTRING("123456789", 0, 999) -- "123456789"
SUBSTRING("123456789", 1, 5)   -- "12345"
```

TRIM

Trims leading or trailing characters from a string. The default character to remove is a space (' ').

Syntax

```
TRIM ( [[LEADING | TRAILING | BOTH remove_chars] FROM] string )
```

Parameters

string

The target string that the function operates on.

`LEADING | TRAILING | BOTH`

This parameter indicates whether to trim leading or trailing characters, or both leading and trailing characters.

`remove_chars`

The set of characters to remove. *`remove_chars`* can be a string with a length > 1. This function returns the string with any character from *`remove_chars`* found at the beginning or end of the string that was removed.

Examples

```
TRIM('       foobar         ')              -- 'foobar'
TRIM('      \tfoobar\t          ')           -- '\tfoobar\t'
TRIM(LEADING FROM '       foobar         ')  -- 'foobar         '
TRIM(TRAILING FROM '       foobar         ') -- '       foobar'
TRIM(BOTH FROM '       foobar         ')     -- 'foobar'
TRIM(BOTH '12' FROM '1112211foobar22211122') -- 'foobar'
```

UPPER

Given a string, UPPER converts all lowercase characters to uppercase characters. Any non-lowercased characters remain unchanged.

Syntax

```
UPPER ( string )
```

Parameters

`string`

The target string that the function operates on.

Examples

```
UPPER('AbCdEfG!@#$') -- 'ABCDEFG!@#$'
```

Working with directory buckets

Directory buckets organize data hierarchically into directories as opposed to the flat storage structure of general purpose buckets. There aren't prefix limits for directory buckets, and individual directories can scale horizontally.

You can create up to 100 directory buckets in each of your AWS accounts, with no limit on the number of objects that you can store in a bucket. Your bucket quota is applied to each Region in your AWS account. If your application requires increasing this limit, contact Support.

Important

Directory buckets in Availability Zones that have no request activity for a period of at least 90 days transition to an inactive state. While in an inactive state, a directory bucket is temporarily inaccessible for reads and writes. Inactive buckets retain all storage, object metadata, and bucket metadata. Existing storage charges apply to inactive buckets. If you make an access request to an inactive bucket, the bucket transitions to an active state, typically within a few minutes. During this transition period, reads and writes return an HTTP `503 (Service Unavailable)` error code. This doesn't apply to buckets in Local Zones.

There are several types of Amazon S3 buckets. Before creating a bucket, make sure that you choose the bucket type that best fits your application and performance requirements. For more information about the various bucket types and the appropriate use cases for each, see Buckets.

The following topics provide information about directory buckets. For more information about general purpose buckets, see General purpose buckets overview.

For more information about directory buckets, see the following topics.

- Directory bucket names
- Directories
- Key names
- Access management

Directory bucket names

A directory bucket name consists of a base name that you provide and a suffix that contains the ID of the Zone (Availability Zone or Local Zone) that your bucket is located in. Directory bucket names must use the following format and follow the naming rules for directory buckets:

```
bucket-base-name--zone-id--x-s3
```

For example, the following directory bucket name contains the Availability Zone ID `usw2-az1`:

```
bucket-base-name--usw2-az1--x-s3
```

For more information, see Directory bucket naming rules.

Directories

Directory buckets organize data hierarchically into directories as opposed to the flat sorting structure of general purpose buckets.

With a hierarchical namespace, the delimiter in the object key is important. The only supported delimiter is a forward slash (/). Directories are determined by delimiter boundaries. For example, the object key `dir1/dir2/file1.txt` results in the directories `dir1/` and `dir2/` being automatically created, and the object `file1.txt` being added to the `/dir2` directory in the path `dir1/dir2/file1.txt`.

The directory bucket indexing model returns unsorted results for the `ListObjectsV2` API operation. If you need to limit your results to a subsection of your bucket, you can specify a subdirectory path in the `prefix` parameter, for example, `prefix=dir1/`.

Key names

For directory buckets, subdirectories that are common to multiple object keys are created with the first object key. Additional object keys for the same subdirectory use the previously created subdirectory. This model gives you flexibility in choosing object keys that are best suited to the application, with equal support for sparse and dense directories.

Access management

Directory buckets have all S3 Block Public Access settings enabled by default at the bucket level. S3 Object Ownership is set to bucket owner enforced and access control lists (ACLs) are disabled. These settings can't be modified.

By default, users don't have permissions for directory buckets. To grant access permissions for directory buckets, you can use IAM to create users, groups, or roles and attach permissions to those identities. For more information, see Authorizing Regional endpoint API operations with IAM.

You can also control access to directory buckets through access points. Access points simplify managing data access at scale for shared datasets in Amazon S3. Access points are unique hostnames you create to enforce distinct permissions and network controls for all requests made through an access point. For more information, see Managing access to shared datasets in directory buckets with access points.

Directory buckets quotas

Quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account. The following are the quotas for directory buckets. For more information on quotas in Amazon S3, see Amazon S3 quotas.

Name	Default	Adjustable	Description
Directory buckets	Each Account: 100	Yes	The number of Amazon S3 directory buckets that you can create in an account.
Read TPS per directory bucket	Each directory bucket: up to 200,000 read TPS	To request a quota increase, contact Support.	The number of GET/HEAD requests per second per directory bucket.
Write TPS per directory bucket	Each directory bucket: up to 100,000 write TPS	To request a quota increase, contact Support.	The number of PUT/DELETE requests per second per directory bucket.

Creating and using directory buckets

For more information about working with directory buckets, see the following topics.

- Use cases for directory buckets
- Differences for directory buckets
- Networking for directory buckets
- Directory bucket naming rules
- Viewing directory bucket properties
- Managing directory bucket policies
- Emptying a directory bucket
- Deleting a directory bucket
- Listing directory buckets
- Determining whether you can access a directory bucket
- Working with objects in a directory bucket
- Security for directory buckets
- Managing access to shared datasets in directory buckets with access points
- Optimizing directory bucket performance
- Developing with directory buckets
- Using tags with S3 directory buckets
- Resilience testing in S3 Express One Zone

Use cases for directory buckets

Directory buckets support bucket creation in the following bucket location types: Availability Zone or Local Zone.

For low latency use cases, you can create a directory bucket in a single Availability Zone to store data. Directory buckets in Availability Zones support the S3 Express One Zone storage class. S3 Express One Zone storage class is recommended if your application is performance sensitive and benefits from single-digit millisecond PUT and GET latencies. To learn more about creating directory buckets in Availability Zones, see High performance workloads.

For data residency use cases, you can create a directory bucket in a single AWS Dedicated Local Zone (DLZ) to store data. Directory buckets in Local Zones support the S3 One Zone-Infrequent Access (S3 One Zone-IA; Z-IA) storage class. To learn more about creating directory buckets in Local Zones, see Data residency workloads.

Topics

- High performance workloads
- Data residency workloads

High performance workloads

S3 Express One Zone

You can use Amazon S3 Express One Zone for high-performance workloads. S3 Express One Zone is the first S3 storage class where you can select a single Availability Zone with the option to co-locate your object storage with your compute resources which provides the highest possible access speed. Objects in S3 Express One Zone are stored in directory buckets located in Availability Zones. For more information on directory buckets, see Directory buckets.

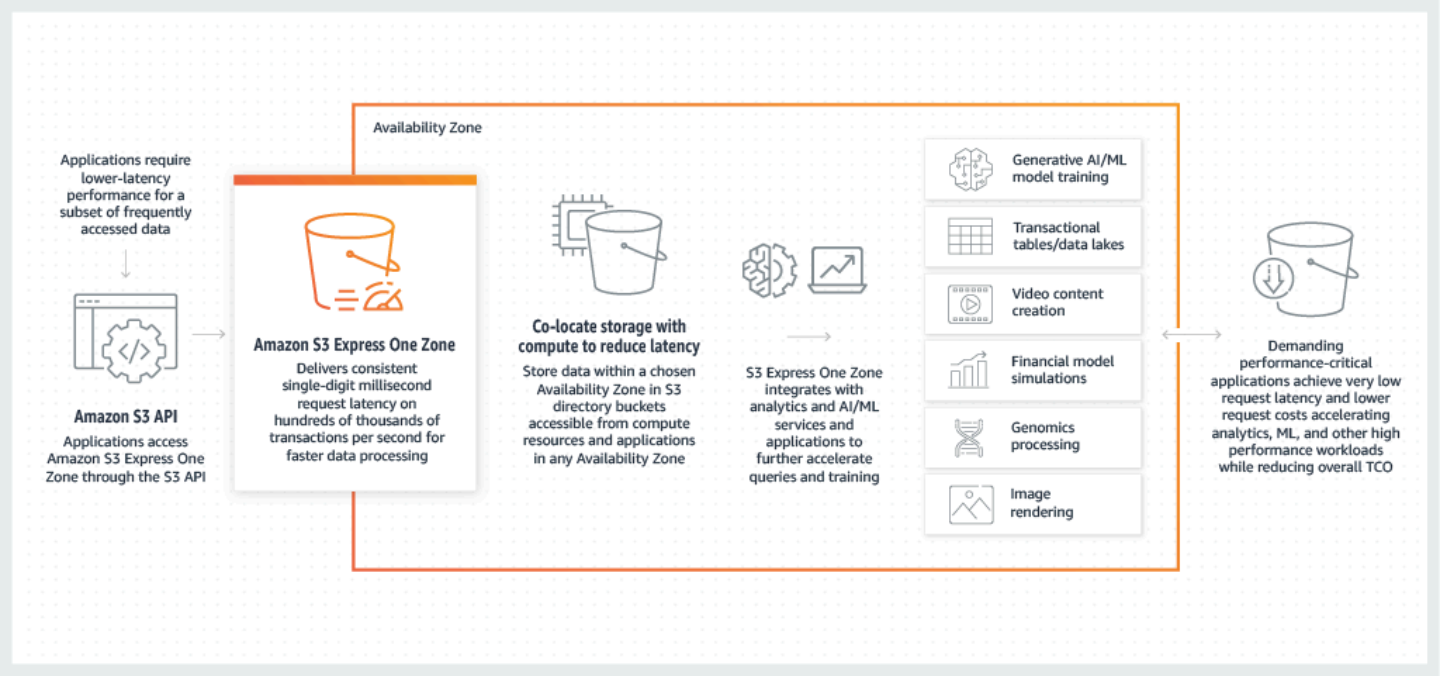
Amazon S3 Express One Zone is a high-performance, single-zone Amazon S3 storage class that is purpose-built to deliver consistent, single-digit millisecond data access for your most latency-sensitive applications. S3 Express One Zone is the lowest latency cloud-object storage class available today, with data access speeds up to 10x faster and with request costs 80 percent lower than S3 Standard. Applications can benefit immediately from requests being completed up to an order of magnitude faster. S3 Express One Zone provides similar performance elasticity as other S3 storage classes. S3 Express One Zone is used for workloads or performance-critical applications that require consistent single-digit millisecond latency.

As with other Amazon S3 storage classes, you don't need to plan or provision capacity or throughput requirements in advance. You can scale your storage up or down, based on need, and access your data through the Amazon S3 API.

The Amazon S3 Express One Zone storage class is designed for 99.95 percent availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed to handle concurrent device failures by quickly detecting and repairing any lost redundancy. If the existing device encounters a failure, S3 Express One Zone

automatically shifts requests to new devices within an Availability Zone. This redundancy helps ensure uninterrupted access to your data within an Availability Zone.

S3 Express One Zone is ideal for any application where it's important to minimize the latency required to access an object. Such applications can be human-interactive workflows, like video editing, where creative professionals need responsive access to content from their user interfaces. S3 Express One Zone also benefits analytics and machine learning workloads that have similar responsiveness requirements from their data, especially workloads with lots of smaller accesses or large numbers of random accesses. S3 Express One Zone can be used with other AWS services to support analytics and artificial intelligence and machine learning (AI/ML) workloads, such as Amazon EMR, Amazon SageMaker AI, and Amazon Athena.



For the directory buckets that use the S3 Express One Zone storage class, data is stored across multiple devices within a single Availability Zone but doesn't store data redundantly across Availability Zones. When you create a directory bucket to use the S3 Express One Zone storage class, we recommend that you specify an AWS Region and an Availability Zone that's local to your Amazon EC2, Amazon Elastic Kubernetes Service, or Amazon Elastic Container Service (Amazon ECS) compute instances to optimize performance.

When using S3 Express One Zone, you can interact with your directory bucket in a virtual private cloud (VPC) by using a gateway VPC endpoint. With a gateway endpoint, you can access S3 Express One Zone directory buckets from your VPC without an internet gateway or NAT device for your VPC, and at no additional cost.

You can use many of the same Amazon S3 API operations and features with directory buckets that you use with general purpose buckets and other storage classes. These include Mountpoint for Amazon S3, server-side encryption with Amazon S3 managed keys (SSE-S3), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), S3 Batch Operations, and S3 Block Public Access. You can access S3 Express One Zone by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, and the Amazon S3 REST API.

For more information about S3 Express One Zone, see the following topics.

- Overview
- Features of S3 Express One Zone
- Related services
- Next steps

Overview

To optimize performance and reduce latency, S3 Express One Zone introduces the following new concepts.

Availability Zones

The Amazon S3 Express One Zone storage class is designed for 99.95 percent availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed to handle concurrent device failures by quickly detecting and repairing any lost redundancy. If the existing device encounters a failure, S3 Express One Zone automatically shifts requests to new devices within an Availability Zone. This redundancy helps ensure uninterrupted access to your data within an Availability Zone.

An Availability Zone is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. When you create a directory bucket, you choose the Availability Zone and AWS Region where your bucket will be located.

Single Availability Zone

When you create a directory bucket, you choose the Availability Zone and AWS Region.

Directory buckets use the S3 Express One Zone storage class, which is built to be used by performance-sensitive applications. S3 Express One Zone is the first S3 storage class where you can

select a single Availability Zone with the option to co-locate your object storage with your compute resources, which provides the highest possible access speed.

With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed for 99.95 percent availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. For more information, see Availability Zones

Endpoints and gateway VPC endpoints

Bucket-management API operations for directory buckets are available through a Regional endpoint and are referred to as Regional endpoint API operations. Examples of Regional endpoint API operations are `CreateBucket` and `DeleteBucket`. After you create a directory bucket, you can use Zonal endpoint API operations to upload and manage the objects in your directory bucket. Zonal endpoint API operations are available through a Zonal endpoint. Examples of Zonal endpoint API operations are `PutObject` and `CopyObject`.

You can access S3 Express One Zone from your VPC by using gateway VPC endpoints. After you create a gateway endpoint, you can add it as a target in your route table for traffic destined from your VPC to S3 Express One Zone. As with Amazon S3, there is no additional charge for using gateway endpoints. For more information about how to configure gateway VPC endpoints, see Networking for directory buckets

Session-based authorization

With S3 Express One Zone, you authenticate and authorize requests through a new session-based mechanism that is optimized to provide the lowest latency. You can use `CreateSession` to request temporary credentials that provide low-latency access to your bucket. These temporary credentials are scoped to a specific S3 directory bucket. Session tokens are used only with Zonal (object-level) operations (with the exception of CopyObject). For more information, see Authorizing Zonal endpoint API operations with `CreateSession`.

The supported AWS SDKs for S3 Express One Zone handle session establishment and refreshment on your behalf. To protect your sessions, temporary security credentials expire after 5 minutes. After you download and install the AWS SDKs and configure the necessary AWS Identity and Access Management (IAM) permissions, you can immediately start using API operations.

Features of S3 Express One Zone

The following S3 features are available for S3 Express One Zone. For a complete list of supported API operationss and unsupported features, see Differences for directory buckets.

Access management and security

You can use the following features to audit and manage access. By default, directory buckets are private and can be accessed only by users who are explicitly granted access. Unlike general purpose buckets, which can set the access control boundary at the bucket, prefix, or object tag level, the access control boundary for directory buckets is set only at the bucket level. For more information, see Authorizing Regional endpoint API operations with IAM.

- S3 Block Public Access – All S3 Block Public Access settings are enabled by default at the bucket level. This default setting can't be modified.
- S3 Object Ownership (bucket owner enforced by default) – Access control lists (ACLs) are not supported for directory buckets. Directory buckets automatically use the bucket owner enforced setting for S3 Object Ownership. Bucket owner enforced means that ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. This default setting can't be modified.
- AWS Identity and Access Management (IAM) – IAM helps you securely control access to your directory buckets. You can use IAM to grant access to bucket management (Regional) API operations and object management (Zonal) API operations through the `s3express:CreateSession` action. For more information, see Authorizing Regional endpoint API operations with IAM. Unlike object-management actions, bucket management actions cannot be cross-account. Only the bucket owner can perform those actions.
- Bucket policies – Use IAM-based policy language to configure resource-based permissions for your directory buckets. You can also use IAM to control access to the `CreateSession` API operation, which allows you to use the Zonal, or object management, API operations. You can grant same-account or cross-account access to Zonal API operations. For more information about S3 Express One Zone permissions and policies, see Authorizing Regional endpoint API operations with IAM.
- IAM Access Analyzer for S3 – Evaluate and monitor your access policies to make sure that the policies provide only the intended access to your S3 resources.

Logging and monitoring

S3 Express One Zone uses the following S3 logging and monitoring tools that you can use to monitor and control how your resources are being used:

- Amazon CloudWatch metrics – Monitor your AWS resources and applications by using CloudWatch to collect and track metrics. S3 Express One Zone uses the same CloudWatch

namespace as other Amazon S3 storage classes (`AWS/S3`) and supports daily storage metrics for directory buckets: `BucketSizeBytes` and `NumberOfObjects`. For more information, see Monitoring metrics with Amazon CloudWatch.

- AWS CloudTrail logs – AWS CloudTrail is an AWS service that helps you implement operational and risk auditing, governance, and compliance of your AWS account by recording the actions taken by a user, role, or an AWS service. For S3 Express One Zone, CloudTrail captures Regional endpoint API operations (for example, `CreateBucket` and `PutBucketPolicy`) as management events and Zonal API operations (for example, `GetObject` and `PutObject`) as data events. These events include actions taken in the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, and AWS API operations. For more information, see Logging with AWS CloudTrail for S3 Express One Zone.

> **Note**
>
> Amazon S3 server access logs aren't supported with S3 Express One Zone.

Object management

You can manage your object storage by using the Amazon S3 console, AWS SDKs, and AWS CLI. The following features are available for object management with S3 Express One Zone:

- S3 Batch Operations – Use Batch Operations to perform bulk operations on objects in directory buckets, for example, **Copy** and **Invoke AWS Lambda function**. For example, you can use Batch Operations to copy objects between directory buckets and general purpose buckets. With Batch Operations, you can manage billions of objects at scale with a single S3 request by using the AWS SDKs or AWS CLI or a few clicks in the Amazon S3 console.
- Import – After you create a directory bucket, you can populate your bucket with objects by using the import feature in the Amazon S3 console. Import is a streamlined method for creating Batch Operations jobs to copy objects from general purpose buckets to directory buckets.

AWS SDKs and client libraries

You can manage your object storage by using the AWS SDKs and client libraries.

- Mountpoint for Amazon S3 – Mountpoint for Amazon S3 is an open source file client that delivers high-throughput access, lowering compute costs for data lakes on Amazon S3.

Mountpoint for Amazon S3 translates local file system API calls to S3 object API calls like `GET` and `LIST`. It is ideal for read-heavy data lake workloads that process petabytes of data and need the high elastic throughput provided by Amazon S3 to scale up and down across thousands of instances.

- S3A – S3A is a recommended Hadoop-compatible interface for accessing data stores in Amazon S3. S3A replaces the S3N Hadoop file system client.
- PyTorch on AWS – PyTorch on AWS is an open source deep-learning framework that makes it easier to develop machine learning models and deploy them to production.
- AWS SDKs – You can use the AWS SDKs when developing applications with Amazon S3. The AWS SDKs simplify your programming tasks by wrapping the underlying Amazon S3 REST API. For more information about using the AWS SDKs with S3 Express One Zone, see the section called "AWS SDKs".

Encryption and data protection

Objects in S3 Express One Zone are automatically encrypted by server-side encryption with Amazon S3 managed keys (SSE-S3). S3 Express One Zone also supports server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS). S3 Express One Zone doesn't support server-side encryption with customer-provided encryption keys (SSE-C), or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS). For more information, see Data protection and encryption.

S3 Express One Zone offers you the option to choose the checksum algorithm that is used to validate your data during upload or download. You can select one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms: CRC32, CRC32C, SHA-1, and SHA-256. MD5-based checksums are not supported with the S3 Express One Zone storage class.

For more information, see S3 additional checksum best practices.

AWS Signature Version 4 (SigV4)

S3 Express One Zone uses AWS Signature Version 4 (SigV4). SigV4 is a signing protocol used to authenticate requests to Amazon S3 over HTTPS. S3 Express One Zone signs requests by using AWS Sigv4. For more information, see Authenticating Requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.

Strong consistency

S3 Express One Zone provides strong read-after-write consistency for PUT and DELETE requests of objects in your directory buckets in all AWS Regions. For more information, see Amazon S3 data consistency model.

Related services

You can use the following AWS services with the S3 Express One Zone storage class to support your specific low-latency use case.

- Amazon Elastic Compute Cloud (Amazon EC2) – Amazon EC2 provides secure and scalable computing capacity in the AWS Cloud. Using Amazon EC2 lessens your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.
- AWS Lambda – Lambda is a compute service that lets you run code without provisioning or managing servers. You configure notification settings on a bucket, and grant Amazon S3 permission to invoke a function on the function's resource-based permissions policy.
- Amazon Elastic Kubernetes Service (Amazon EKS) – Amazon EKS is a managed service that eliminates the need to install, operate, and maintain your own Kubernetes control plane on AWS. Kubernetes is an open source system that automates the management, scaling, and deployment of containerized applications.
- Amazon Elastic Container Service (Amazon ECS) – Amazon ECS is a fully managed container orchestration service that helps you easily deploy, manage, and scale containerized applications.
- AWS Key Management Service (AWS KMS) – AWS Key Management Service (AWS KMS) is an AWS managed service that makes it easy for you to create and control the encryption keys that are used to encrypt your data. The AWS KMS keys that you create in AWS KMS are protected by FIPS 140-2 validated hardware security modules (HSM). To use or manage your KMS keys, you interact with AWS KMS.
- Amazon Athena – Athena is an interactive query service that makes it easy to analyze data directly in Amazon S3 by using standard SQL. You can also use Athena to interactively run data analytics by using Apache Spark without having to plan for, configure, or manage resources. When you run Apache Spark applications on Athena, you submit Spark code for processing and receive the results directly.

- Amazon SageMaker Training – Review the options for training models with Amazon SageMaker, including built-in algorithms, custom algorithms, libraries, and models from the AWS Marketplace.
- AWS Glue – AWS Glue is a serverless data-integration service that makes it easy for analytics users to discover, prepare, move, and integrate data from multiple sources. You can use AWS Glue for analytics, machine learning, and application development. AWS Glue also includes additional productivity and data-ops tooling for authoring, running jobs, and implementing business workflows.
- Amazon EMR – Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data.
- AWSCloudTrail – AWSCloudTrail is s an AWS service that helps you enable operational and risk auditing, governance, and compliance of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail. Events include actions taken in the AWS Management Console, AWS Command Line Interface, and AWS SDKs and APIs.
- AWS CloudFormation – is a service that helps you model and set up your AWS resources so that you can spend less time managing those resources and more time focusing on your applications that run in AWS. You create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and CloudFormation takes care of provisioning and configuring those resources for you. You don't need to individually create and configure AWS resources and figure out what's dependent on what; CloudFormation handles that.

Next steps

For more information about working with the S3 Express One Zone storage class and directory buckets, see the following topics:

- Tutorial: Getting started with S3 Express One Zone
- S3 Express One Zone Availability Zones and Regions
- Networking for directory buckets in an Availability Zone
- Creating directory buckets in an Availability Zone
- Regional and Zonal endpoints for directory buckets in an Availability Zone
- Optimizing S3 Express One Zone performance

Tutorial: Getting started with S3 Express One Zone

Amazon S3 Express One Zone is the first S3 storage class where you can select a single Availability Zone with the option to co-locate your object storage with your compute resources which provides the highest possible access speed. Data in S3 Express One Zone is stored in directory buckets located in Availability Zones. For more information on directory buckets, see Directory buckets.

S3 Express One Zone is ideal for any application where it's critical to minimize request latency. Such applications can be human-interactive workflows, like video editing, where creative professionals need responsive access to content from their user interfaces. S3 Express One Zone also benefits analytics and machine learning workloads that have similar responsiveness requirements from their data, especially workloads with a lot of smaller accesses or a large numbers of random accesses. S3 Express One Zone can be used with other AWS services such as Amazon EMR, Amazon Athena, AWS Glue Data Catalog and Amazon SageMaker Model Training to support analytics, artificial intelligence and machine learning (AI/ML) workloads,. You can work with the S3 Express One Zone storage class and directory buckets by using the Amazon S3 console, AWS SDKs, AWS Command Line Interface (AWS CLI), and Amazon S3 REST API. For more information, see What is S3 Express One Zone? and How is S3 Express One Zone different?.
This is an S3 Express One Zone workflow diagram.

Objective

In this tutorial, you will learn how to create a gateway endpoint, create and attach an IAM policy, create a directory bucket and then use the Import action to populate your directory bucket with objects currently stored in your general purpose bucket. Alternatively, you can manually upload objects to your directory bucket.

Topics

- Prerequisites
- Step 1: Configure a gateway VPC endpoint to reach S3 Express One Zone directory buckets
- Step 2: Create a S3 Express One Zone directory bucket
- Step 3: Importing data into a S3 Express One Zone directory bucket
- Step 4: Manually upload objects to your S3 Express One Zone directory bucket
- Step 5: Empty your S3 Express One Zone directory bucket
- Step 6: Delete your S3 Express One Zone directory bucket
- Next steps

Prerequisites

Before you start this tutorial, you must have an AWS account that you can sign in to as an AWS Identity and Access Management (IAM) user with correct permissions.

Substeps

- Create an AWS account
- Create an IAM user in your AWS account (console)
- Create an IAM policy and attach it to an IAM user or role (console)

Create an AWS account

To complete this tutorial, you need an AWS account. When you sign up for AWS, your AWS account is automatically signed up for all services in AWS, including Amazon S3. You are charged only for the services that you use. For more information about pricing, see S3 pricing.

Create an IAM user in your AWS account (console)

AWS Identity and Access Management (IAM) is an AWS service that helps administrators securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to access objects and use directory buckets in S3 Express One Zone. You can use IAM for no additional charge.

By default, users don't have permissions to access directory buckets and perform S3 Express One Zone operations. To grant access permissions for directory buckets and S3 Express One Zone operations, you can use IAM to create users or roles and attach permissions to those identities. For more information about how to create an IAM user, see Creating IAM users (console) in the *IAM User Guide*. For more information about how to create an IAM role, see Creating a role to delegate permissions to an IAM user in the *IAM User Guide*.

For simplicity, this tutorial creates and uses an IAM user. After completing this tutorial, remember to Delete the IAM user. For production use, we recommend that you follow the Security best practices in IAM in the *IAM User Guide*. A best practice requires human users to use federation with an identity provider to access AWS with temporary credentials. Another best practice is to require workloads to use temporary credentials with IAM roles to access AWS. To learn more about using AWS IAM Identity Center to create users with temporary credentials, see Getting started in the *AWS IAM Identity Center User Guide*.

> **Warning**
>
> IAM users have long-term credentials, which presents a security risk. To help mitigate this risk, we recommend that you provide these users with only the permissions they require to perform the task and that you remove these users when they are no longer needed.

Create an IAM policy and attach it to an IAM user or role (console)

By default, users don't have permissions for directory buckets and S3 Express One Zone operations. To grant access permissions for directory buckets, you can use IAM to create users, groups, or roles and attach permissions to those identities. Directory buckets are the only resource that you can include in bucket policies or IAM identity policies for S3 Express One Zone access.

To use Regional endpoint API operations (bucket-level or control plane operations) with S3 Express One Zone, you use the IAM authorization model, which doesn't involve session management. Permissions are granted for actions individually. To use Zonal endpoint API operations (object-level or data plane operations), you use `CreateSession` to create and manage sessions that are optimized for low-latency authorization of data requests. To retrieve and use a session token, you must allow the `s3express:CreateSession` action for your directory bucket in an identity-based policy or a bucket policy. If you're accessing S3 Express One Zone in the Amazon S3 console, through the AWS Command Line Interface (AWS CLI), or by using the AWS SDKs, S3 Express One Zone creates a session on your behalf. For more information, see `CreateSession` authorization and AWS Identity and Access Management (IAM) for S3 Express One Zone .

To create an IAM policy and attach the policy to an IAM user (or role)

1. Sign in to the AWS Management Console and open the IAM Management Console.
2. In the navigation pane, choose **Policies**.
3. Choose **Create Policy**.
4. Select **JSON**.
5. Copy the policy below into the **Policy editor** window. Before you can create directory buckets or use S3 Express One Zone, you must grant the necessary permissions to your AWS Identity and Access Management (IAM) role or users. This example policy allows access to the `CreateSession` API operation (for use with other Zonal or object-level API operations) and all of the Regional endpoint (bucket-level) API operations. This policy allows the `CreateSession` API operation for use with all directory buckets, but the Regional

endpoint API operations are allowed only for use with the specified directory bucket. To use this example policy, replace the *user input placeholders* with your own information.

JSON

```
{
     "Version":"2012-10-17",
     "Statement": [
         {
             "Sid": "AllowAccessRegionalEndpointAPIs",
             "Effect": "Allow",
             "Action": [
                 "s3express:DeleteBucket",
                 "s3express:DeleteBucketPolicy",
                 "s3express:CreateBucket",
                 "s3express:PutBucketPolicy",
                 "s3express:GetBucketPolicy",
                 "s3express:ListAllMyDirectoryBuckets"
             ],

             "Resource": "arn:aws:s3express:us-
east-1:111122223333:bucket/amzn-s3-demo-bucket--usw2-az1--x-s3/*"
         },
         {
             "Sid": "AllowCreateSession",
             "Effect": "Allow",
             "Action": "s3express:CreateSession",
             "Resource": "*"
         }
     ]
 }
```

6. Choose **Next**.
7. Name the policy.

> **Note**
>
> Bucket tags are not supported for S3 Express One Zone.

8. Select **Create policy**.

9. Now that you've created an IAM policy, you can attach it to an IAM user. In the navigation pane, choose **Policies**.
10. In the **search bar**, enter the name of your policy.
11. From the **Actions** menu, select **Attach**.
12. Under **Filter by Entity Type**, select **IAM users** or **Roles**.
13. In the **search field**, type the name of the user or role you wish to use.
14. Choose **Attach Policy**.

Step 1: Configure a gateway VPC endpoint to reach S3 Express One Zone directory buckets

You can access both Zonal and Regional API operations through gateway virtual private cloud (VPC) endpoints. Gateway endpoints can allow traffic to reach S3 Express One Zone without traversing a NAT Gateway. We strongly recommend using gateway endpoints as they provide the most optimal networking path when working with S3 Express One Zone. You can access S3 Express One Zone directory buckets from your VPC without an internet gateway or NAT device for your VPC, and at no additional cost. Use the following procedure to configure a gateway endpoint that connects to S3 Express One Zone storage class objects and directory buckets.

To access S3 Express One Zone, you use Regional and Zonal endpoints that are different from standard Amazon S3 endpoints. Depending on the Amazon S3 API operation that you use, either a Zonal or Regional endpoint is required. For a complete list of supported API operations by endpoint type, see API operations supported by S3 Express One Zone. You must access both Zonal and Regional endpoints through a gateway virtual private cloud (VPC) endpoint.

Use the following procedure to create a gateway endpoint that connects to S3 Express One Zone storage class objects and directory buckets.

To configure a gateway VPC endpoint

1. Open the Amazon VPC Console at https://console.aws.amazon.com/vpc/.
2. In the side navigation pane under **Virtual private cloud**, choose **Endpoints**.
3. Choose **Create endpoint**.
4. Create a name for your endpoint.
5. For **Service category**, choose **AWS services**.
6. Under **Services**, search using the filter **Type=Gateway** and then choose the option button next to **com.amazonaws.*region*.s3express**.

7. For **VPC**, choose the VPC in which to create the endpoint.
8. For **Route tables**, choose the route table on your Local Zone to be used by the endpoint. After the endpoint is created, a route record will be added to the route table that you select in this step.
9. For **Policy**, choose **Full access** to allow all operations by all principals on all resources over the VPC endpoint. Otherwise, choose **Custom** to attach a VPC endpoint policy that controls the principals' permissions to perform actions on resources over the VPC endpoint.
10. For **IP address type**, choose from the following options:

    - **IPv4** – Assign IPv4 addresses to the endpoint network interfaces. This option is supported only if all selected subnets have IPv4 address ranges and the service accepts IPv4 requests.
    - **IPv6** – Assign IPv6 addresses to the endpoint network interfaces. This option is supported only if all selected subnets are IPv6 only subnets and the service accepts IPv6 requests.
    - **Dualstack** – Assign both IPv4 and IPv6 addresses to the endpoint network interfaces. This option is supported only if all selected subnets have both IPv4 and IPv6 address ranges and the service accepts both IPv4 and IPv6 requests.
11. (Optional) To add a tag, choose **Add new tag**, and enter the tag key and the tag value.
12. Choose **Create endpoint**.

After creating a gateway endpoint, you can use Regional API endpoints and Zonal API endpoints to access Amazon S3 Express One Zone storage class objects and directory buckets.

Step 2: Create a S3 Express One Zone directory bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

> **Note**
>
> To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.

3. In the left navigation pane, choose **Directory buckets**.
4. Choose **Create bucket**. The **Create bucket** page opens.
5. Under **General configuration**, view the AWS Region where your bucket will be created.

   Under **Bucket type**, choose **Directory**.

   > **Note**
   >
   > - If you've chosen a Region that doesn't support directory buckets, the **Bucket type** option disappears, and the bucket type defaults to a general purpose bucket. To create a directory bucket, you must choose a supported Region. For a list of Regions that support directory buckets and the Amazon S3 Express One Zone storage class, see the section called "S3 Express One Zone Availability Zones and Regions".
   > - After you create the bucket, you can't change the bucket type.

   > **Note**
   >
   > The Availability Zone can't be changed after the bucket is created.

6. For **Availability Zone**, choose a Availability Zone local to your compute services. For a list of Availability Zones that support directory buckets and the S3 Express One Zone storage class, see the section called "S3 Express One Zone Availability Zones and Regions".

   Under **Availability Zone**, select the check box to acknowledge that in the event of an Availability Zone outage, your data might be unavailable or lost.

   > **Important**
   >
   > Although directory buckets are stored across multiple devices within a single Availability Zone, directory buckets don't store data redundantly across Availability Zones.

7. For **Bucket name**, enter a name for your directory bucket.

   The following naming rules apply for directory buckets.

   - Be unique within the chosen Zone (AWS Availability Zone or AWS Local Zone).

- Name must be between 3 (min) and 63 (max) characters long, including the suffix.
- Consists only of lowercase letters, numbers and hyphens (-).
- Begin and end with a letter or number.
- Must include the following suffix: `--`*`zone-id`*`--x-s3`.
- Bucket names must not start with the prefix `xn--`.
- Bucket names must not start with the prefix `sthree-`.
- Bucket names must not start with the prefix `sthree-configurator`.
- Bucket names must not start with the prefix `amzn-s3-demo-`.
- Bucket names must not end with the suffix `-s3alias`. This suffix is reserved for access point alias names. For more information, see Access point aliases.
- Bucket names must not end with the suffix `--ol-s3`. This suffix is reserved for Object Lambda Access Point alias names. For more information, see How to use a bucket-style alias for your S3 bucket Object Lambda Access Point.
- Bucket names must not end with the suffix `.mrap`. This suffix is reserved for Multi-Region Access Point names. For more information, see Rules for naming Amazon S3 Multi-Region Access Points.

A suffix is automatically added to the base name that you provide when you create a directory bucket using the console. This suffix includes the Availability Zone ID of the Availability Zone that you chose.

After you create the bucket, you can't change its name. For more information about naming buckets, see General purpose bucket naming rules.

> **Important**
>
> Do not include sensitive information, such as account numbers, in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

8. Under **Object Ownership**, the **Bucket owner enforced** setting is automatically enabled, and all access control lists (ACLs) are disabled. For directory buckets, ACLs can't be enabled.

   Bucket owner enforced (default) – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the general purpose bucket. ACLs no longer

affect access permissions to data in the S3 general purpose bucket. The bucket uses policies exclusively to define access control.

9. Under **Block Public Access settings for this bucket**, all Block Public Access settings for your directory bucket are automatically enabled. These settings can't be modified for directory buckets. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

10. To configure default encryption, under **Encryption type**, choose one of the following:

    - **Server-side encryption with Amazon S3 managed key (SSE-S3)**
    - **Server-side encryption with AWS Key Management Service key (SSE-KMS)**

    For more information about using Amazon S3 server-side encryption to encrypt your data, see Data protection and encryption.

    > **Important**
    >
    > If you use the SSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quota of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.
    >
    > When you enable default encryption, you might need to update your bucket policy. For more information, see Using SSE-KMS encryption for cross-account operations.

11. If you chose **Server-side encryption with Amazon S3 managed keys (SSE-S3)**, under **Bucket Key**, **Enabled** appears. S3 Bucket Keys are always enabled when you configure your directory bucket to use default encryption with SSE-S3. S3 Bucket Keys are always enabled for GET and PUT operations in a directory bucket and can't be disabled. S3 Bucket Keys aren't supported, when you copy SSE-KMS encrypted objects from general purpose buckets to directory buckets, from directory buckets to general purpose buckets, or between directory buckets, through CopyObject, UploadPartCopy, the Copy operation in Batch Operations, or the import jobs. In this case, Amazon S3 makes a call to AWS KMS every time a copy request is made for a KMS-encrypted object.

    S3 Bucket Keys lower the cost of encryption by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

12. If you chose **Server-side encryption with AWS Key Management Service key (SSE-KMS)**, under **AWS KMS key**, specify your AWS Key Management Service key in one of the following ways or create a new key.

    - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from **Available AWS KMS keys**.

      Only your customer managed keys appear in this list. The AWS managed key (`aws/s3`) isn't supported in directory buckets. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.

    - To enter the KMS key ARN or alias, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN or alias in **AWS KMS key ARN**.
    - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

      For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

    Important

    - Your SSE-KMS configuration can only support 1 customer managed key per directory bucket for the lifetime of the bucket. The AWS managed key (`aws/s3`) isn't supported. Also, after you specify a customer managed key for SSE-KMS, you can't override the customer managed key for the bucket's SSE-KMS configuration.

      You can identify the customer managed key you specified for the bucket's SSE-KMS configuration, in the following way:

      - You make a `HeadObject` API operation request to find the value of `x-amz-server-side-encryption-aws-kms-key-id` in your response.

      To use a new customer managed key for your data, we recommend copying your existing objects to a new directory bucket with a new customer managed key.

    - You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that is not listed, you must enter your KMS key ARN. If you want to use a KMS key that is owned by a different account, you must first have permission to use the key and then you must enter the KMS key ARN. For more information on cross account permissions for KMS keys, see Creating KMS keys that

other accounts can use in the *AWS Key Management Service Developer Guide*. For more information on SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS) for new object uploads in directory buckets.

- When you use an AWS KMS key for server-side encryption in directory buckets, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

For more information about using AWS KMS with Amazon S3, see Using server-side encryption with AWS KMS keys (SSE-KMS) in directory buckets.

13. Choose **Create bucket**. After creating the bucket, you can add files and folders to the bucket. For more information, see the section called "Working with objects in a directory bucket".

Step 3: Importing data into a S3 Express One Zone directory bucket

To complete this step, you must have a general purpose bucket that contains objects and is located in the same AWS Region as your directory bucket.

After you create a directory bucket in Amazon S3, you can populate the new bucket with data by using the Import action in the Amazon S3 console. Import simplifies copying data into directory buckets by letting you choose a prefix or a general purpose bucket to Import data from without having to specify all of the objects to copy individually. Import uses S3 Batch Operations which copies the objects in the selected prefix or general purpose bucket. You can monitor the progress of the Import copy job through the S3 Batch Operations job details page.

To use the Import action

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region associated with the Availability Zone in which your directory bucket is located.
3. In the left navigation pane, choose **Directory buckets**.
4. Choose the option button next to the name of the bucket that you want to import objects into.

5. Choose **Import**.
6. For **Source**, enter the general purpose bucket (or bucket path including prefix) that contains the objects that you want to import. To choose an existing general purpose bucket from a list, choose **Browse S3**.
7. In the **Permissions** section, you can choose to have an IAM role auto-generated. Alternatively, you can select an IAM role from a list, or directly enter an IAM role ARN.

   - To allow Amazon S3 to create a new IAM role on your behalf, choose **Create new IAM role**.

     > **Note**
     >
     > If your source objects are encrypted with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), don't choose the **Create new IAM role** option. Instead, specify an existing IAM role that has the `kms:Decrypt` permission.
     > Amazon S3 will use this permission to decrypt your objects. During the import process, Amazon S3 will then re-encrypt those objects by using server-side encryption with Amazon S3 managed keys (SSE-S3).

   - To choose an existing IAM role from a list, choose **Choose from existing IAM roles**.
   - To specify an existing IAM role by entering its Amazon Resource Name (ARN), choose **Enter IAM role ARN**, then enter the ARN in the corresponding field.
8. Review the information that's displayed in the **Destination** and **Copied object settings** sections. If the information in the **Destination** section is correct, choose **Import** to start the copy job.

   The Amazon S3 console displays the status of your new job on the **Batch Operations** page. For more information about the job, choose the option button next to the job name, and then on the **Actions** menu, choose **View details**. To open the directory bucket that the objects will be imported into, choose **View import destination**.

Step 4: Manually upload objects to your S3 Express One Zone directory bucket

You can also manually upload objects to your directory bucket.

To manually upload objects

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region associated with the Availability Zone in which your directory bucket is located.
3. In the left navigation pane, choose **Directory buckets**.
4. Choose the name of the bucket that you want to upload your folders or files to.

   > **Note**
   >
   > If you chose the same directory bucket that you used in previous steps of this tutorial, your directory bucket will contain the objects that were uploaded from the Import tool. Notice that these objects are now stored in the S3 Express One Zone storage class.

5. In the **Objects** list, choose **Upload**.
6. On the **Upload** page, do one of the following:

   - Drag and drop files and folders to the dotted upload area.
   - Choose **Add files** or **Add folder**, choose the files or folders to upload, and then choose **Open** or **Upload**.

7. Under **Checksums**, choose the **Checksum function** that you want to use.

   > **Note**
   >
   > We recommend using CRC32 and CRC32C for the best performance with the S3 Express One Zone storage class. For more information, see S3 additional checksum best practices.

   (Optional) If you're uploading a single object that's less than 16 MB in size, you can also specify a pre-calculated checksum value. When you provide a pre-calculated value, Amazon S3 compares it with the value that it calculates by using the selected checksum function. If the values don't match, the upload won't start.

8. The options in the **Permissions** and **Properties** sections are automatically set to default settings and can't be modified. Block Public Access is automatically enabled, and S3 Versioning and S3 Object Lock can't be enabled for directory buckets.

   (Optional) If you want to add metadata in key-value pairs to your objects, expand the **Properties** section, and then in the **Metadata** section, choose **Add metadata**.

9. To upload the listed files and folders, choose **Upload**.

   Amazon S3 uploads your objects and folders. When the upload is finished, you see a success message on the **Upload: status** page.

   You have successfully created a directory bucket and uploaded objects to your bucket.

Step 5: Empty your S3 Express One Zone directory bucket

You can empty your Amazon S3 directory bucket by using the Amazon S3 console.

To empty a directory bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region associated with the Availability Zone in which your directory bucket is located.
3. In the left navigation pane, choose **Directory buckets**.
4. Choose the option button next to the name of the bucket that you want to empty, and then choose **Empty**.
5. On the **Empty bucket** page, confirm that you want to empty the bucket by entering **`permanently delete`** in the text field, and then choose **Empty**.
6. Monitor the progress of the bucket emptying process on the **Empty bucket: status** page.

Step 6: Delete your S3 Express One Zone directory bucket

After you empty your directory bucket and abort all in-progress multipart uploads, you can delete your bucket by using the Amazon S3 console.

To delete a directory bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region associated with the Availability Zone in which your directory bucket is located.
3. In the left navigation pane, choose **Directory buckets**.
4. In the **Directory buckets** list, choose the option button next to the bucket that you want to delete.
5. Choose **Delete**.
6. On the **Delete bucket** page, enter the name of the bucket in the text field to confirm the deletion of your bucket.

   > **Important**
   >
   > Deleting a directory bucket can't be undone.

7. To delete your directory bucket, choose **Delete bucket**.

Next steps

In this tutorial, you have learned how to create a directory bucket and use the S3 Express One Zone storage class. After completing this tutorial, you can explore related AWS services to use with the S3 Express One Zone storage class.

You can use the following AWS services with the S3 Express One Zone storage class to support your specific low-latency use case.

- Amazon Elastic Compute Cloud (Amazon EC2) – Amazon EC2 provides secure and scalable computing capacity in the AWS Cloud. Using Amazon EC2 lessens your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need, configure security and networking, and manage storage.
- AWS Lambda – Lambda is a compute service that lets you run code without provisioning or managing servers. You configure notification settings on a bucket, and grant Amazon S3 permission to invoke a function on the function's resource-based permissions policy.

- Amazon Elastic Kubernetes Service (Amazon EKS) – Amazon EKS is a managed service that eliminates the need to install, operate, and maintain your own Kubernetes control plane on AWS. Kubernetes is an open-source system that automates the management, scaling, and deployment of containerized applications.
- Amazon Elastic Container Service (Amazon ECS) – Amazon ECS is a fully managed container orchestration service that helps you easily deploy, manage, and scale containerized applications.
- Amazon EMR – Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark on AWS to process and analyze vast amounts of data.
- Amazon Athena – Athena is an interactive query service that makes it easy to analyze data directly in Amazon S3 by using standard SQL. You can also use Athena to interactively run data analytics by using Apache Spark without having to plan for, configure, or manage resources. When you run Apache Spark applications on Athena, you submit Spark code for processing and receive the results directly.
- AWS Glue Data Catalog – AWS Glue is a serverless data-integration service that makes it easy for analytics users to discover, prepare, move, and integrate data from multiple sources. You can use AWS Glue for analytics, machine learning, and application development. AWS Glue Data Catalog is a centralized repository that stores metadata about your organization's data sets. It acts as an index to the location, schema, and run-time metrics of your data sources.
- Amazon SageMaker Runtime Model Training – Amazon SageMaker Runtime is a fully managed machine learning service. With SageMaker Runtime, data scientists and developers can quickly and easily build and train machine learning models, and then directly deploy them into a production-ready hosted environment.

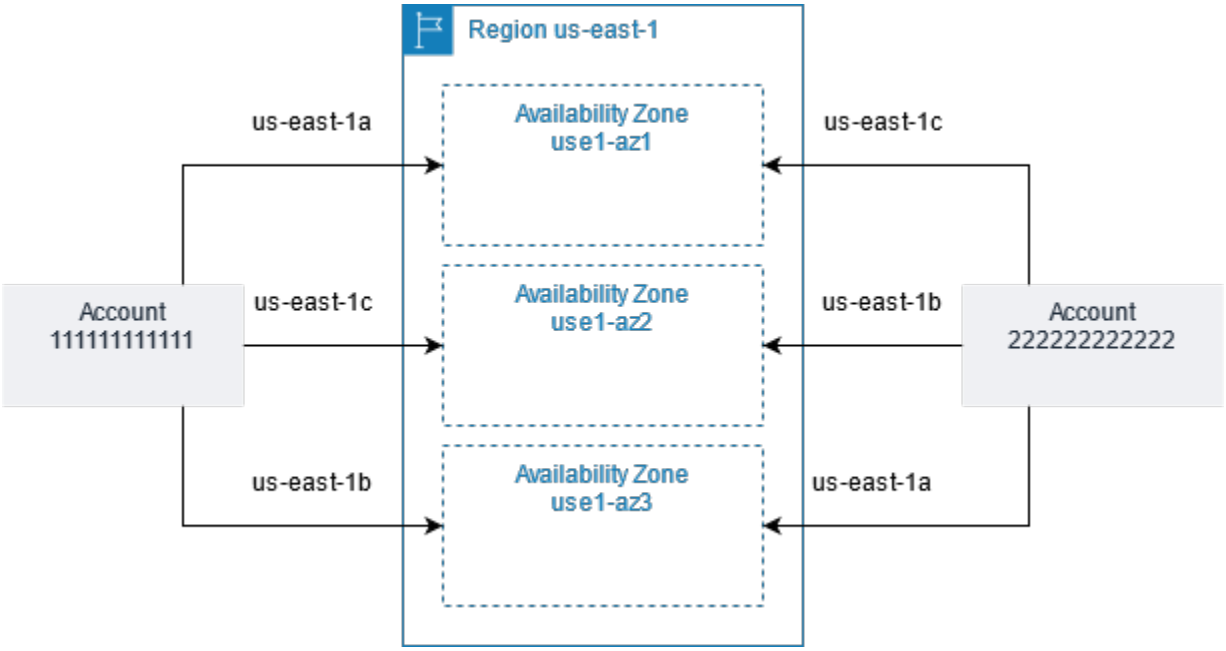
For more information on S3 Express One Zone, see What is S3 Express One Zone? and How is S3 Express One Zone different?.

S3 Express One Zone Availability Zones and Regions

An Availability Zone is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. To optimize low-latency retrievals, objects in the Amazon S3 Express One Zone storage class are redundantly stored in S3 directory buckets in a single Availability Zone that's local to your compute workload. When you create a directory bucket, you choose the Availability Zone and AWS Region where your bucket will be located.

AWS maps the physical Availability Zones randomly to the Availability Zone names for each AWS account. This approach helps to distribute resources across the Availability Zones in an AWS Region, instead of resources likely being concentrated in the first Availability Zone for each Region. As a result, the Availability Zone `us-east-1a` for your AWS account might not represent the same physical location as `us-east-1a` for a different AWS account. For more information, see Regions and Availability Zones in the *Amazon EC2 User Guide*.

To coordinate Availability Zones across accounts, you must use the *AZ ID*, which is a unique and consistent identifier for an Availability Zone. For example, `use1-az1` is an AZ ID for the `us-east-1` Region and it has the same physical location in every AWS account. The following illustration shows how the AZ IDs are the same for every account, even though the Availability Zone names might be mapped differently for each account.



With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed for 99.95 percent availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. For more information, see Availability Zones

The following table shows the S3 Express One Zone supported Regions and Availability Zones.

Region name	Region code	Availability Zone ID				
US East (N. Virginia)	`us-east-1`	`use1-az4`				

Region name	Region code	Availability Zone ID				
		use1-az5				
		use1-az6				
US East (Ohio)	us-east-2	use2-az1				
		use2-az2				
US West (N. California)	us-west-1	usw1-az1				
US West (Oregon)	us-west-2	usw2-az1				
		usw2-az3				
		usw2-az4				
Asia Pacific (Hong Kong)	ap-east-1	ape1-az1				
Asia Pacific (Jakarta)	ap-southeast-3	apse3-az1				
Asia Pacific (Mumbai)	ap-south-1	aps1-az1				
		aps1-az3				
Asia Pacific (Seoul)	ap-northeast-2	apne2-az3				
Asia Pacific (Singapore)	ap-southeast-1	apse1-az1				
Asia Pacific (Sydney)	ap-southeast-2	apse2-az1				
Asia Pacific (Tokyo)	ap-northeast-1	apne1-az1				

Region name	Region code	Availability Zone ID				
		apne1-az4				
Europe (Frankfurt)	eu-central-1	euc1-az1				
		euc1-az2				
Europe (Ireland)	eu-west-1	euw1-az1				
		euw1-az3				
Europe (London)	eu-west-2	euw2-az1				
Europe (Paris)	eu-west-3	euw3-az1				
Europe (Stockholm)	eu-north-1	eun1-az1				
		eun1-az2				
		eun1-az3				
Canada (Central)	ca-central-1	cac1-az1				
South America (São Paulo)	sa-east-1	sae1-az1				
Africa (Cape Town)	af-south-1	afs1-az1				

Networking for directory buckets in an Availability Zone

To reduce the amount of time your packets spend on the network, configure your virtual private cloud (VPC) with a gateway endpoint to access directory buckets in Availability Zones while keeping traffic within the AWS network, and at no additional cost.

Topics

- Endpoints for directory buckets in Availability Zones

- Configuring VPC gateway endpoints

Endpoints for directory buckets in Availability Zones

The following table shows the Regional and Zonal API endpoints that are available for each Region and Availability Zone.

Configuring VPC gateway endpoints

Use the following procedure to create a gateway endpoint that connects to Amazon S3 Express One Zone storage class objects and directory buckets.

To configure a gateway VPC endpoint

1. Open the Amazon VPC Console.
2. In the navigation pane, choose **Endpoints**.
3. Choose **Create endpoint**.
4. Create a name for your endpoint.
5. For **Service category**, choose **AWS services**.
6. For **Services**, add the filter **Type=Gateway** and then choose the option button next to **com.amazonaws.*region*.s3express**.
7. For **VPC**, choose the VPC in which to create the endpoint.
8. For **Route tables**, choose the route table in your VPC to be used by the endpoint. After the endpoint is created, a route record will be added to the route table that you select in this step.
9. For **Policy**, choose **Full access** to allow all operations by all principals on all resources over the VPC endpoint. Otherwise, choose **Custom** to attach a VPC endpoint policy that controls the principals' permissions to perform actions on resources over the VPC endpoint.
10. For **IP address type**, choose from the following options:

    - **IPv4** – Assign IPv4 addresses to the endpoint network interfaces. This option is supported only if all selected subnets have IPv4 address ranges and the service accepts IPv4 requests.
    - **IPv6** – Assign IPv6 addresses to the endpoint network interfaces. This option is supported only if all selected subnets are IPv6 only subnets and the service accepts IPv6 requests.
    - **Dualstack** – Assign both IPv4 and IPv6 addresses to the endpoint network interfaces. This option is supported only if all selected subnets have both IPv4 and IPv6 address ranges and the service accepts both IPv4 and IPv6 requests.

11. (Optional) To add a tag, choose **Add new tag**, and enter the tag key and the tag value.
12. Choose **Create endpoint**.

After creating a gateway endpoint, you can use Regional API endpoints and Zonal API endpoints to access Amazon S3 Express One Zone storage class objects and directory buckets.

To learn more about gateway VPC endpoints, see Gateway endpoints in the *AWS PrivateLink Guide*. For the data residency use cases, we recommend enabling access to your buckets only from your VPC using gateway VPC endpoints. When access is restricted to a VPC or a VPC endpoint, you can access the objects through the AWS Management Console, the REST API, AWS CLI, and AWS SDKs.

> **Note**
>
> To restrict access to a VPC or a VPC endpoint using the AWS Management Console, you must use the AWS Management Console Private Access. For more information, see AWS Management Console Private Access in the *AWS Management Console guide*AWS Management Console guide.

Creating directory buckets in an Availability Zone

To start using the Amazon S3 Express One Zone storage class, you create a directory bucket. The S3 Express One Zone storage class can be used only with directory buckets. The S3 Express One Zone storage class supports low-latency use cases and provides faster data processing within a single Availability Zone. If your application is performance sensitive and benefits from single-digit millisecond PUT and GET latencies, we recommend creating a directory bucket so that you can use the S3 Express One Zone storage class.

There are two types of Amazon S3 buckets, general purpose buckets and directory buckets. You should choose the bucket type that best fits your application and performance requirements. General purpose buckets are the original S3 bucket type. General purpose buckets are recommended for most use cases and access patterns and allow objects stored across all storage classes, except S3 Express One Zone. For more information about general purpose buckets, see General purpose buckets overview.

Directory buckets use the S3 Express One Zone storage class, which is designed to be used for workloads or performance-critical applications that require consistent single-digit millisecond

latency. S3 Express One Zone is the first S3 storage class where you can select a single Availability Zone with the option to co-locate your object storage with your compute resources, which provides the highest possible access speed. When you create a directory bucket, you can optionally specify an AWS Region and an Availability Zone that's local to your Amazon EC2, Amazon Elastic Kubernetes Service, or Amazon Elastic Container Service (Amazon ECS) compute instances to optimize performance.

With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. S3 Express One Zone is designed for 99.95 percent availability within a single Availability Zone and is backed by the Amazon S3 Service Level Agreement. For more information, see Availability Zones

Directory buckets organize data hierarchically into directories, as opposed to the flat storage structure of general purpose buckets. There aren't prefix limits for directory buckets, and individual directories can scale horizontally.

For more information about directory buckets, see Working with directory buckets.

Directory bucket names

Directory bucket names must follow this format and comply with the rules for directory bucket naming:

```
bucket-base-name--zone-id--x-s3
```

For example, the following directory bucket name contains the Availability Zone ID `usw2-az1`:

```
bucket-base-name--usw2-az1--x-s3
```

For more information about directory bucket naming rules, see Directory bucket naming rules.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

> **Note**
>
> To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.

3. In the left navigation pane, choose **Directory buckets**.
4. Choose **Create bucket**. The **Create bucket** page opens.
5. Under **General configuration**, view the AWS Region where your bucket will be created.

   Under **Bucket type**, choose **Directory**.

> **Note**
>
> - If you've chosen a Region that doesn't support directory buckets, the **Bucket type** option disappears, and the bucket type defaults to a general purpose bucket. To create a directory bucket, you must choose a supported Region. For a list of Regions that support directory buckets and the Amazon S3 Express One Zone storage class, see the section called "S3 Express One Zone Availability Zones and Regions".
> - After you create the bucket, you can't change the bucket type.

> **Note**
>
> The Availability Zone can't be changed after the bucket is created.

6. For **Availability Zone**, choose a Availability Zone local to your compute services. For a list of Availability Zones that support directory buckets and the S3 Express One Zone storage class, see the section called "S3 Express One Zone Availability Zones and Regions".

   Under **Availability Zone**, select the check box to acknowledge that in the event of an Availability Zone outage, your data might be unavailable or lost.

> **Important**
>
> Although directory buckets are stored across multiple devices within a single Availability Zone, directory buckets don't store data redundantly across Availability Zones.

7. For **Bucket name**, enter a name for your directory bucket.

   The following naming rules apply for directory buckets.

   - Be unique within the chosen Zone (AWS Availability Zone or AWS Local Zone).
   - Name must be between 3 (min) and 63 (max) characters long, including the suffix.
   - Consists only of lowercase letters, numbers and hyphens (-).
   - Begin and end with a letter or number.
   - Must include the following suffix: `--`*`zone-id`*`--x-s3`.
   - Bucket names must not start with the prefix `xn--`.
   - Bucket names must not start with the prefix `sthree-`.
   - Bucket names must not start with the prefix `sthree-configurator`.
   - Bucket names must not start with the prefix `amzn-s3-demo-`.
   - Bucket names must not end with the suffix `-s3alias`. This suffix is reserved for access point alias names. For more information, see Access point aliases.
   - Bucket names must not end with the suffix `--ol-s3`. This suffix is reserved for Object Lambda Access Point alias names. For more information, see How to use a bucket-style alias for your S3 bucket Object Lambda Access Point.
   - Bucket names must not end with the suffix `.mrap`. This suffix is reserved for Multi-Region Access Point names. For more information, see Rules for naming Amazon S3 Multi-Region Access Points.

   A suffix is automatically added to the base name that you provide when you create a directory bucket using the console. This suffix includes the Availability Zone ID of the Availability Zone that you chose.

   After you create the bucket, you can't change its name. For more information about naming buckets, see General purpose bucket naming rules.

> **Important**
>
> Do not include sensitive information, such as account numbers, in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

8. Under **Object Ownership**, the **Bucket owner enforced** setting is automatically enabled, and all access control lists (ACLs) are disabled. For directory buckets, ACLs can't be enabled.

   Bucket owner enforced (default) – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the general purpose bucket. ACLs no longer affect access permissions to data in the S3 general purpose bucket. The bucket uses policies exclusively to define access control.

9. Under **Block Public Access settings for this bucket**, all Block Public Access settings for your directory bucket are automatically enabled. These settings can't be modified for directory buckets. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

10. To configure default encryption, under **Encryption type**, choose one of the following:

    - **Server-side encryption with Amazon S3 managed key (SSE-S3)**
    - **Server-side encryption with AWS Key Management Service key (SSE-KMS)**

    For more information about using Amazon S3 server-side encryption to encrypt your data, see Data protection and encryption.

    > **Important**
    >
    > If you use the SSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quota of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.
    >
    > When you enable default encryption, you might need to update your bucket policy. For more information, see Using SSE-KMS encryption for cross-account operations.

11. If you chose **Server-side encryption with Amazon S3 managed keys (SSE-S3)**, under **Bucket Key**, **Enabled** appears. S3 Bucket Keys are always enabled when you configure your directory bucket to use default encryption with SSE-S3. S3 Bucket Keys are always enabled for GET and

PUT operations in a directory bucket and can't be disabled. S3 Bucket Keys aren't supported, when you copy SSE-KMS encrypted objects from general purpose buckets to directory buckets, from directory buckets to general purpose buckets, or between directory buckets, through CopyObject, UploadPartCopy, the Copy operation in Batch Operations, or the import jobs. In this case, Amazon S3 makes a call to AWS KMS every time a copy request is made for a KMS-encrypted object.

S3 Bucket Keys lower the cost of encryption by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

12. If you chose **Server-side encryption with AWS Key Management Service key (SSE-KMS)**, under **AWS KMS key**, specify your AWS Key Management Service key in one of the following ways or create a new key.

    - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from **Available AWS KMS keys**.

      Only your customer managed keys appear in this list. The AWS managed key (`aws/s3`) isn't supported in directory buckets. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
    - To enter the KMS key ARN or alias, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN or alias in **AWS KMS key ARN**.
    - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

      For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

    Important

    - Your SSE-KMS configuration can only support 1 customer managed key per directory bucket for the lifetime of the bucket. The AWS managed key (`aws/s3`) isn't supported. Also, after you specify a customer managed key for SSE-KMS, you can't override the customer managed key for the bucket's SSE-KMS configuration.

      You can identify the customer managed key you specified for the bucket's SSE-KMS configuration, in the following way:

- You make a `HeadObject` API operation request to find the value of `x-amz-server-side-encryption-aws-kms-key-id` in your response.

To use a new customer managed key for your data, we recommend copying your existing objects to a new directory bucket with a new customer managed key.

- You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that is not listed, you must enter your KMS key ARN. If you want to use a KMS key that is owned by a different account, you must first have permission to use the key and then you must enter the KMS key ARN. For more information on cross account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*. For more information on SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS) for new object uploads in directory buckets.
- When you use an AWS KMS key for server-side encryption in directory buckets, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

For more information about using AWS KMS with Amazon S3, see Using server-side encryption with AWS KMS keys (SSE-KMS) in directory buckets.

13. Choose **Create bucket**. After creating the bucket, you can add files and folders to the bucket. For more information, see the section called "Working with objects in a directory bucket".

Using the AWS SDKs

SDK for Go

This example shows how to create a directory bucket by using the AWS SDK for Go.

Example

```
var bucket = "..."

func runCreateBucket(c *s3.Client) {
    resp, err := c.CreateBucket(context.Background(), &s3.CreateBucketInput{
```

```
        Bucket: &bucket,
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            Location: &types.LocationInfo{
                Name: aws.String("usw2-az1"),
                Type: types.LocationTypeAvailabilityZone,
            },
            Bucket: &types.BucketInfo{
                DataRedundancy: types.DataRedundancySingleAvailabilityZone,
                Type:           types.BucketTypeDirectory,
            },
        },
    })
    var terr *types.BucketAlreadyOwnedByYou
    if errors.As(err, &terr) {
        fmt.Printf("BucketAlreadyOwnedByYou: %s\n", aws.ToString(terr.Message))
        fmt.Printf("noop...\n")
        return
    }
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("bucket created at %s\n", aws.ToString(resp.Location))
}
```

SDK for Java 2.x

This example shows how to create an directory bucket by using the AWS SDK for Java 2.x.

Example

```
public static void createBucket(S3Client s3Client, String bucketName) {

    //Bucket name format is {base-bucket-name}--{az-id}--x-s3
    //example: doc-example-bucket--usw2-az1--x-s3 is a valid name for a directory
 bucket created in
    //Region us-west-2, Availability Zone 2

    CreateBucketConfiguration bucketConfiguration =
 CreateBucketConfiguration.builder()
             .location(LocationInfo.builder()
                     .type(LocationType.AVAILABILITY_ZONE)
                     .name("usw2-az1").build()) //this must match the Region and
 Availability Zone in your bucket name
```

```
            .bucket(BucketInfo.builder()
                   .type(BucketType.DIRECTORY)
                   .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                   .build()).build();
    try {

            CreateBucketRequest bucketRequest =
 CreateBucketRequest.builder().bucket(bucketName).createBucketConfiguration(bucketConfigurat
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            System.out.println(response);
    }

    catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
        }
    }
```

AWS SDK for JavaScript

This example shows how to create a directory bucket by using the AWS SDK for JavaScript.

Example

```
// file.mjs, run with Node.js v16 or higher
// To use with the preview build, place this in a folder
// inside the preview build directory, such as /aws-sdk-js-v3/workspace/

import { S3 } from "@aws-sdk/client-s3";

const region = "us-east-1";
const zone = "use1-az4";
const suffix = `${zone}--x-s3`;

const s3 = new S3({ region });

const bucketName = `...--${suffix}`;

const createResponse = await s3.createBucket(
    { Bucket: bucketName,
      CreateBucketConfiguration: {Location: {Type: "AvailabilityZone", Name: zone},
```

```
      Bucket: { Type: "Directory", DataRedundancy: "SingleAvailabilityZone" }}
    }
   );
```

SDK for .NET

This example shows how to create a directory bucket by using the SDK for .NET.

Example

```
using (var amazonS3Client = new AmazonS3Client())
{
    var putBucketResponse = await amazonS3Client.PutBucketAsync(new PutBucketRequest
    {

       BucketName = "DOC-EXAMPLE-BUCKET--usw2-az1--x-s3",
       PutBucketConfiguration = new PutBucketConfiguration
       {
         BucketInfo = new BucketInfo { DataRedundancy =
 DataRedundancy.SingleAvailabilityZone, Type = BucketType.Directory },
         Location = new LocationInfo { Name = "usw2-az1", Type =
 LocationType.AvailabilityZone }
       }
     }).ConfigureAwait(false);
}
```

SDK for PHP

This example shows how to create a directory bucket by using the AWS SDK for PHP.

Example

```
require 'vendor/autoload.php';

$s3Client = new S3Client([

    'region'      => 'us-east-1',
]);


$result = $s3Client->createBucket([
```

```
    'Bucket' => 'doc-example-bucket--use1-az4--x-s3',
    'CreateBucketConfiguration' => [
        'Location' => ['Name'=> 'use1-az4', 'Type'=> 'AvailabilityZone'],
        'Bucket' => ["DataRedundancy" => "SingleAvailabilityZone" ,"Type" =>
 "Directory"]   ],
]);
```

SDK for Python (Boto3)

This example shows how to create a directory bucket by using the AWS SDK for Python (Boto3).

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def create_bucket(s3_client, bucket_name, availability_zone):
    '''
    Create a directory bucket in a specified Availability Zone

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to create; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :param availability_zone: String; Availability Zone ID to create the bucket in,
 for example, 'usw2-az1'
    :return: True if bucket is created, else False
    '''

    try:
        bucket_config = {
                'Location': {
                    'Type': 'AvailabilityZone',
                    'Name': availability_zone
                },
                'Bucket': {
                    'Type': 'Directory',
                    'DataRedundancy': 'SingleAvailabilityZone'
                }
            }
        s3_client.create_bucket(
            Bucket = bucket_name,
            CreateBucketConfiguration = bucket_config
```

```
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    availability_zone = 'usw2-az1'
    s3_client = boto3.client('s3', region_name = region)
    create_bucket(s3_client, bucket_name, availability_zone)
```

SDK for Ruby

This example shows how to create an directory bucket by using the AWS SDK for Ruby.

Example

```
s3 = Aws::S3::Client.new(region:'us-west-2')
s3.create_bucket(
  bucket: "bucket_base_name--az_id--x-s3",
  create_bucket_configuration: {
    location: { name: 'usw2-az1', type: 'AvailabilityZone' },
    bucket: { data_redundancy: 'SingleAvailabilityZone', type: 'Directory' }
  }
)
```

Using the AWS CLI

This example shows how to create a directory bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

When you create a directory bucket you must provide configuration details and use the following naming convention: *`bucket-base-name`*`--`*`zone-id`*`--x-s3`

```
aws s3api create-bucket
--bucket bucket-base-name--zone-id--x-s3
--create-bucket-configuration 'Location={Type=AvailabilityZone,Name=usw2-
az1},Bucket={DataRedundancy=SingleAvailabilityZone,Type=Directory}'
```

```
--region us-west-2
```

For more information, see create-bucket in the AWS Command Line Interface.

Regional and Zonal endpoints for directory buckets in an Availability Zone

To access your objects and directory buckets stored in S3 Express One Zone, you use gateway VPC endpoints. Directory buckets use Regional and Zonal API endpoints. Depending on the Amazon S3 API operation that you use, either a Regional or Zonal endpoint is required. There is no additional charge for using gateway endpoints.

Bucket-level (or control plane) API operations are available through Regional endpoints and are referred to as Regional endpoint API operations. Examples of Regional endpoint API operations are `CreateBucket` and `DeleteBucket`.

When you create directory buckets that are stored in S3 Express One Zone, you choose the Availability Zone where your bucket will be located. You can use Zonal endpoint API operations to upload and manage the objects in your directory bucket.

Object-level (or data plane) API operations are available through Zonal endpoints and are referred to as Zonal endpoint API operations. Examples of Zonal endpoint API operations are `CreateSession` and `PutObject`.

Optimizing S3 Express One Zone performance

Amazon S3 Express One Zone is a high-performance, single Availability Zone (AZ) S3 storage class that's purpose-built to deliver consistent, single-digit millisecond data access for your most latency-sensitive applications. S3 Express One Zone is the first S3 storage class that gives you the option to co-locate high-performance object storage and AWS compute resources, such as Amazon Elastic Compute Cloud, Amazon Elastic Kubernetes Service, and Amazon Elastic Container Service, within a single Availability Zone. Co-locating your storage and compute resources optimizes compute performance and costs and provides increased data-processing speed.

S3 Express One Zone provides similar performance elasticity to other S3 storage classes, but with consistent single-digit millisecond first-byte read and write request latencies—up to 10x faster than S3 Standard. S3 Express One Zone is designed from the ground up to support burst throughput up to very high aggregate levels. The S3 Express One Zone storage class uses a custom-built architecture to optimize for performance and deliver consistently low request latency by storing data on high-performance hardware. The object protocol for S3 Express One Zone has been enhanced to streamline authentication and metadata overhead.

To further reduce latency and support up to 2 million reads and up to 200,000 writes per second, S3 Express One Zone stores data in an Amazon S3 directory bucket. By default, each directory bucket supports up to 200,000 reads and up to 100,000 writes per second. If your workload requires higher than the default TPS limits, you can request an increase through AWS Support.

The combination of high-performance, purpose-built hardware and software that delivers single-digit millisecond data access speed and directory buckets that scale for large numbers of transactions per second makes S3 Express One Zone the best Amazon S3 storage class for request-intensive operations or performance-critical applications.

The following topics describe best practice guidelines and design patterns for optimizing performance with applications that use the S3 Express One Zone storage class.

Topics

- Best practices to optimize S3 Express One Zone performance

Best practices to optimize S3 Express One Zone performance

When building applications that upload and retrieve objects from Amazon S3 Express One Zone, follow our best practice guidelines to optimize performance. To use the S3 Express One Zone storage class, you must create an S3 directory bucket. The S3 Express One Zone storage class isn't supported for use with S3 general purpose buckets.

For performance guidelines for all other Amazon S3 storage classes and S3 general purpose buckets, see Best practices design patterns: optimizing Amazon S3 performance.

For optimal performance and scalability with S3 Express One Zone storage class and directory buckets in high-scale workloads, it's important to understand how directory buckets work differently from general purpose buckets. Then, we provide best practices to align your applications with the way directory buckets work.

How directory buckets work

Amazon S3 Express One Zone storage class can support workloads with up to 2,000,000 GET and up to 200,000 PUT transactions per second (TPS) per directory bucket. With S3 Express One Zone, data is stored in S3 directory buckets in Availability Zones. Objects in directory buckets are accessible within a hierarchical namespace, similar to a file system and in contrast to S3 general purpose buckets that have a flat namespace. Unlike general purpose buckets, directory buckets organize keys hierarchically into directories instead of prefixes. A prefix is a string of characters at

the beginning of the object key name. You can use prefixes to organize your data and manage a flat object storage architecture in general purpose buckets. For more information, see Organizing objects using prefixes.

In directory buckets, objects are organized in a hierarchical namespace using forward slash (/) as the only supported delimiter. When you upload an object with a key like `dir1/dir2/file1.txt`, the directories `dir1/` and `dir2/` are automatically created and managed by Amazon S3. Directories are created during `PutObject` or `CreateMultiPartUpload` operations and automatically removed when they become empty after `DeleteObject` or `AbortMultiPartUpload` operations. There is no upper limit to the number of objects and subdirectories in a directory.

The directories that are created when objects are uploaded to directory buckets can scale instantaneously to reduce the chance of HTTP `503 (Slow Down)` errors. This automatic scaling allows your applications to parallelize read and write requests within and across directories as needed. For S3 Express One Zone, individual directories are designed to support the maximum request rate of a directory bucket. There is no need to randomize key prefixes to achieve optimal performance as the system automatically distributes objects for even load distribution, but as a result, keys are not stored lexicographically in directory buckets. This is in contrast to S3 general purpose buckets where keys that are lexicographically closer are more likely to be co-located on the same server.

For more information about examples of directory bucket operations and directory interactions, see Directory bucket operation and directory interaction examples.

Best practices

Follow the best practices to optimize your directory bucket performance and help your workloads scale over time.

Use directories that contain many entries (objects or subdirectories)

Directory buckets deliver high performance by default for all workloads. For even greater performance optimization with certain operations, consolidating more entries (which are objects or subdirectories) into directories will lead to lower latency and higher request rate:

- Mutating API operations, such as `PutObject`, `DeleteObject`, `CreateMultiPartUpload` and `AbortMultiPartUpload`, achieve optimal performance when implemented with fewer, denser directories containing thousands of entries, rather than with a large number of smaller directories.

- `ListObjectsV2` operations perform better when fewer directories need to be traversed to populate a page of results.

Don't use entropy in prefixes

In Amazon S3 operations, entropy refers to the randomness in prefix naming that helps distribute workloads evenly across storage partitions. However, since directory buckets internally manage load distribution, it's not recommended to use entropy in prefixes for the best performance. This is because for directory buckets, entropy can cause requests to be slower by not reusing the directories that have already been created.

A key pattern such as `$HASH/directory/object` could end up creating many intermediate directories. In the following example, all the `job-1` s are different directories since their parents are different. Directories will be sparse and mutation and list requests will be slower. In this example there are 12 intermediate Directories that all have a single entry.

```
s3://my-bucket/0cc175b9c0f1b6a831c399e269772661/job-1/file1

s3://my-bucket/92eb5ffee6ae2fec3ad71c777531578f/job-1/file2

s3://my-bucket/4a8a08f09d37b73795649038408b5f33/job-1/file3

s3://my-bucket/8277e0910d750195b448797616e091ad/job-1/file4

s3://my-bucket/e1671797c52e15f763380b45e841ec32/job-1/file5

s3://my-bucket/8fa14cdd754f91cc6554c9e71929cce7/job-1/file6
```

Instead, for better performance, we can remove the `$HASH` component and allow `job-1` to become a single directory, improving the density of a directory. In the following example, the single intermediate directory that has 6 entries can lead to better performance, compared with the previous example.

```
s3://my-bucket/job-1/file1

s3://my-bucket/job-1/file2

s3://my-bucket/job-1/file3

s3://my-bucket/job-1/file4
```

```
s3://my-bucket/job-1/file5

s3://my-bucket/job-1/file6
```

This performance advantage occurs because when an object key is initially created and its key name includes a directory, the directory is automatically created for the object. Subsequent object uploads to that same directory do not require the directory to be created, which reduces latency on object uploads to existing directories.

Use a separator other than the delimiter / to separate parts of your key if you don't need the ability to logically group objects during `ListObjectsV2` calls

Since the / delimiter is treated specially for directory buckets, it should be used with intention. While directory buckets do not lexicographically order objects, objects within a directory are still grouped together in `ListObjectsV2` outputs. If you don't need this functionality, you can replace / with another character as a separator to not cause the creation of intermediate directories.

For example, assume the following keys are in a `YYYY/MM/DD/HH/` prefix pattern

```
s3://my-bucket/2024/04/00/01/file1

s3://my-bucket/2024/04/00/02/file2

s3://my-bucket/2024/04/00/03/file3

s3://my-bucket/2024/04/01/01/file4

s3://my-bucket/2024/04/01/02/file5

s3://my-bucket/2024/04/01/03/file6
```

If you don't have the need to group objects by hour or day in `ListObjectsV2` results, but you need to group objects by month, the following key pattern of `YYYY/MM/DD-HH-` will lead to significantly fewer directories and better performance for the `ListObjectsV2` operation.

```
s3://my-bucket/2024/04/00-01-file1

s3://my-bucket/2024/04/00-01-file2
```

```
s3://my-bucket/2024/04/00-01-file3

s3://my-bucket/2024/04/01-02-file4

s3://my-bucket/2024/04/01-02-file5

s3://my-bucket/2024/04/01-02-file6
```

Use delimited list operations where possible

A `ListObjectsV2` request without a `delimiter` performs depth-first recursive traversal of all directories. A `ListObjectsV2` request with a `delimiter` retrieves only entries in the directory specified by the `prefix` parameter, reducing request latency and increasing aggregate keys per second. For directory buckets, use delimited list operations where possible. Delimited lists result in directories being visited fewer times, which leads to more keys per second and lower request latency.

For example, for the following directories and objects in your directory bucket:

```
s3://my-bucket/2024/04/12-01-file1

s3://my-bucket/2024/04/12-01-file2

...

s3://my-bucket/2024/05/12-01-file1

s3://my-bucket/2024/05/12-01-file2

...

s3://my-bucket/2024/06/12-01-file1

s3://my-bucket/2024/06/12-01-file2

...

s3://my-bucket/2024/07/12-01-file1

s3://my-bucket/2024/07/12-01-file2

...
```

For better `ListObjectsV2` performance, use a delimited list to list your subdirectories and objects, if your application's logic allows for it. For example, you can run the following command for the delimited list operation,

```
aws s3api list-objects-v2 --bucket my-bucket --prefix '2024/' --delimiter '/'
```

The output is the list of subdirectories.

```
{
    "CommonPrefixes": [
        {
            "Prefix": "2024/04/"
        },
        {
            "Prefix": "2024/05/"
        },
        {
            "Prefix": "2024/06/"
        },
        {
            "Prefix": "2024/07/"
        }
    ]
}
```

To list each subdirectory with better performance, you can run a command like the following example:

Command:

```
aws s3api list-objects-v2 --bucket my-bucket --prefix '2024/04' --delimiter '/'
```

Output:

```
{
    "Contents": [
        {
            "Key": "2024/04/12-01-file1"
        },
        {
            "Key": "2024/04/12-01-file2"
```

```
        }
    ]
}
```

Co-locate S3 Express One Zone storage with your compute resources

With S3 Express One Zone, each directory bucket is located in a single Availability Zone that you select when you create the bucket. You can get started by creating a new directory bucket in an Availability Zone local to your compute workloads or resources. You can then immediately begin very low-latency reads and writes. Directory buckets are a type of S3 buckets where you can choose the Availability Zone in an AWS Region to reduce latency between compute and storage.

If you access directory buckets across Availability Zones, you may experience slightly increased latency. To optimize performance, we recommend that you access a directory bucket from Amazon Elastic Container Service, Amazon Elastic Kubernetes Service, and Amazon Elastic Compute Cloud instances that are located in the same Availability Zone when possible.

Use concurrent connections to achieve high throughput with objects over 1MB

You can achieve the best performance by issuing multiple concurrent requests to directory buckets to spread your requests over separate connections to maximize the accessible bandwidth. Like general purpose buckets, S3 Express One Zone doesn't have any limits for the number of connections made to your directory bucket. Individual directories can scale performance horizontally and automatically when large numbers of concurrent writes to the same directory are happening.

Individual TCP connections to directory buckets have a fixed upper bound on the number of bytes that can be uploaded or downloaded per second. When objects get larger, request times become dominated by byte streaming rather than transaction processing. To use multiple connections to parallelize the upload or download of larger objects, you can reduce end-to-end latency. If using the `Java 2.x` SDK, you should consider using the S3 Transfer Manager which will take advantage of performance improvements such as the multipart upload API operations and byte-range fetches to access data in parallel.

Use Gateway VPC endpoints

Gateway endpoints provide a direct connection from your VPC to directory buckets, without requiring an internet gateway or a NAT device for your VPC. To reduce the amount of time your packets spend on the network, you should configure your VPC with a gateway VPC endpoint for directory buckets. For more information, see Networking for directory buckets.

Use session authentication and reuse session tokens while they're valid

Directory buckets provide a session token authentication mechanism to reduce latency on performance-sensitive API operations. You can make a single call to `CreateSession` to get a session token which is then valid for all requests in the following 5 minutes. To get the lowest latency in your API calls, make sure to acquire a session token and reuse it for the entire lifetime of that token before refreshing it.

If you use AWS SDKs, SDKs handle the session token refreshes automatically to avoid service interruptions when a session expires. We recommend that you use the AWS SDKs to initiate and manage requests to the `CreateSession` API operation.

For more information about `CreateSession`, see Authorizing Zonal endpoint API operations with `CreateSession`.

Use a CRT-based client

The AWS Common Runtime (CRT) is a set of modular, performant, and efficient libraries written in C and meant to act as the base of the AWS SDKs. The CRT provides improved throughput, enhanced connection management, and faster startup times. The CRT is available through all the AWS SDKs except Go.

For more information on how to configure the CRT for the SDK you use, see AWS Common Runtime (CRT) libraries, Accelerate Amazon S3 throughput with the AWS Common Runtime, Introducing CRT-based S3 client and the S3 Transfer Manager in the AWS SDK for Java 2.x, Using S3CrtClient for Amazon S3 operations, and Configure AWS CRT-based HTTP clients.

Use the latest version of the AWS SDKs

The AWS SDKs provide built-in support for many of the recommended guidelines for optimizing Amazon S3 performance. The SDKs offer a simpler API for taking advantage of Amazon S3 from within an application and are regularly updated to follow the latest best practices. For example, the SDKs automatically retry requests after HTTP `503` errors and handle slow connections responses.

If using the `Java 2.x` SDK, you should consider using the S3 Transfer Manager, which automatically scales connections horizontally to achieve thousands of requests per second using byte-range requests when appropriate. Byte-range requests can improve performance because you can use concurrent connections to S3 to fetch different byte ranges from within the same object. This helps you achieve higher aggregate throughput versus a single whole-object request. So it's important to use the latest version of the AWS SDKs to obtain the latest performance optimization features.

Performance troubleshooting

Are you setting retry requests for latency-sensitive applications?

S3 Express One Zone is purpose-built to deliver consistent levels of high-performance without additional tuning. However, setting aggressive timeout values and retries can further help drive consistent latency and performance. The AWS SDKs have configurable timeout and retry values that you can tune to the tolerances of your specific application.

Are you using AWS Common Runtime (CRT) libraries and optimal Amazon EC2 instance types?

Applications that perform a large number of read and write operations likely need more memory or computing capacity than applications that don't. When launching your Amazon Elastic Compute Cloud (Amazon EC2) instances for your performance-demanding workload, choose instance types that have the amount of these resources that your application needs. S3 Express One Zone high-performance storage is ideally paired with larger and newer instance types with larger amounts of system memory and more powerful CPUs and GPUs that can take advantage of higher-performance storage. We also recommend using the latest versions of the CRT-enabled AWS SDKs, which can better accelerate read and write requests in parallel.

Are you using AWS SDKs for session-based authentication?

With Amazon S3, you can also optimize performance when you're using HTTP REST API requests by following the same best practices that are part of the AWS SDKs. However, with the session-based authorization and authentication mechanism that's used by S3 Express One Zone, we strongly recommend that you use the AWS SDKs to manage `CreateSession` and its managed session token. The AWS SDKs automatically create and refresh tokens on your behalf by using the `CreateSession` API operation. Using `CreateSession` saves on per-request round-trip latency to AWS Identity and Access Management (IAM) to authorize each request.

Directory bucket operation and directory interaction examples

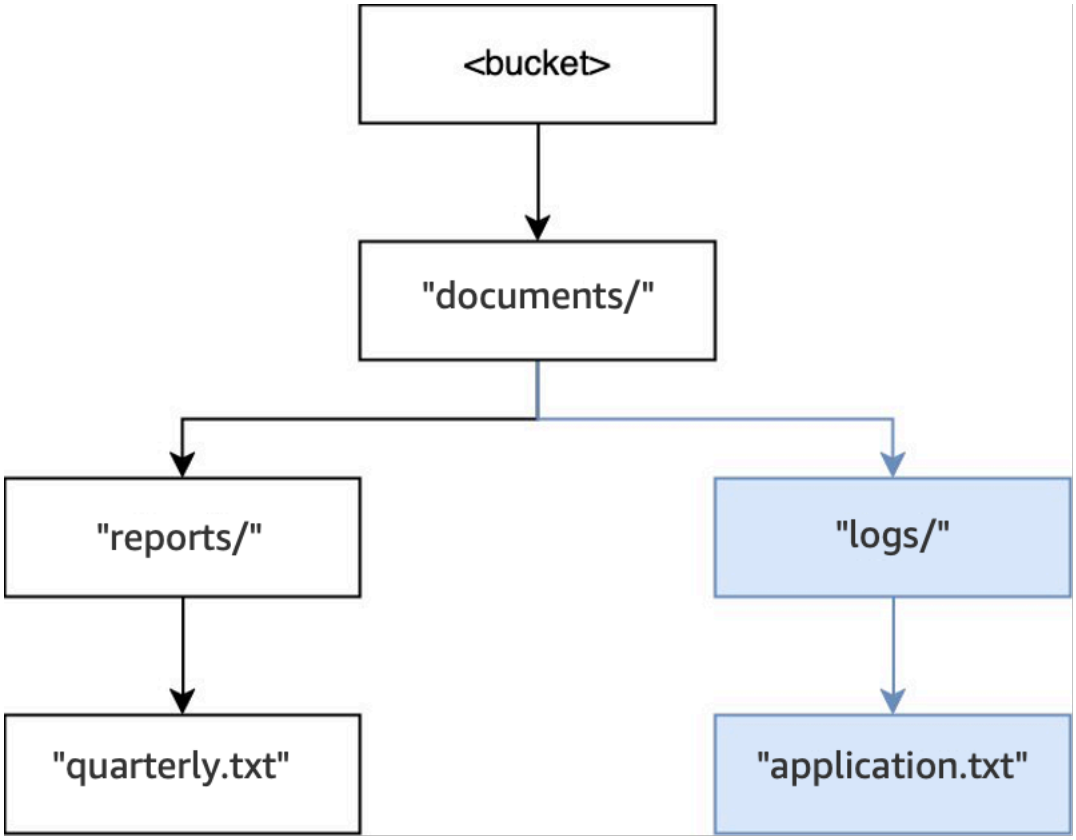
The following shows three examples about how directory buckets work.

Example 1: How S3 `PutObject` requests to a directory bucket interact with directories

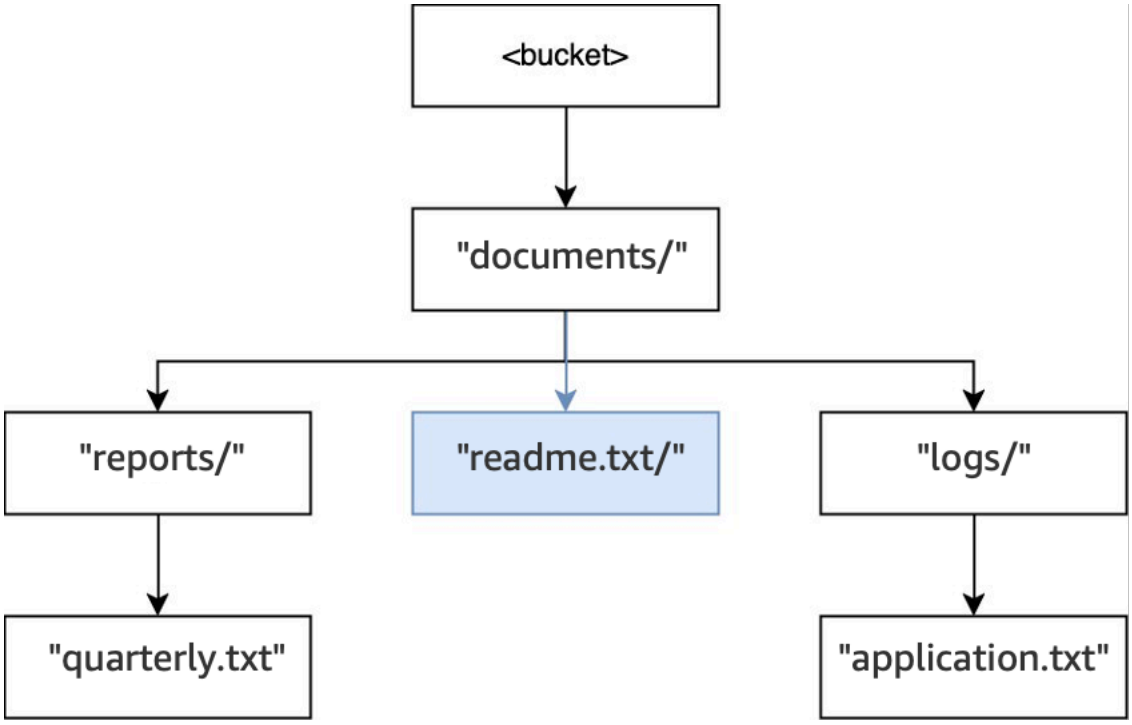
1. When the operation `PUT(<bucket>, "documents/reports/quarterly.txt")` is executed in an empty bucket, the directory `documents/` within the root of the bucket is created, the directory `reports/` within `documents/` is created, and the object `quarterly.txt` within `reports/` is created. For this operation, two directories were created in addition to the object.



2. Then, when another operation `PUT(<bucket>, "documents/logs/application.txt")` is executed, the directory `documents/` already exists, the directory `logs/` within `documents/` doesn't exist and is created, and the object `application.txt` within `logs/` is created. For this operation, only one directory was created in addition to the object.



3. Lastly, when a PUT(<bucket>, "documents/readme.txt") operation is executed, the directory documents/ within the root already exists and the object readme.txt is created. For this operation, no directories are created.



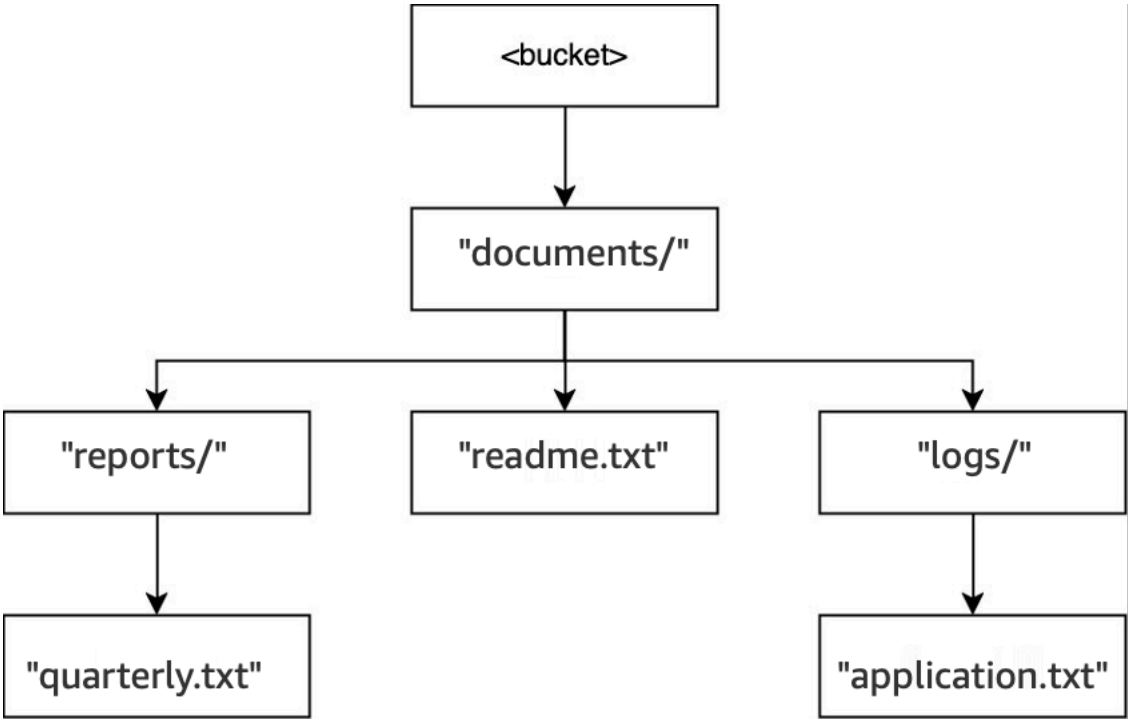
Example 2: How S3 `ListObjectsV2` requests to a directory bucket interact with directories

For the S3 `ListObjectsV2` requests without specifying a delimiter, a bucket is traversed in a depth-first manner. The outputs are returned in a consistent order. However, while this order remains the same between requests, the order is not lexicographic. For the bucket and directories created in the previous example:

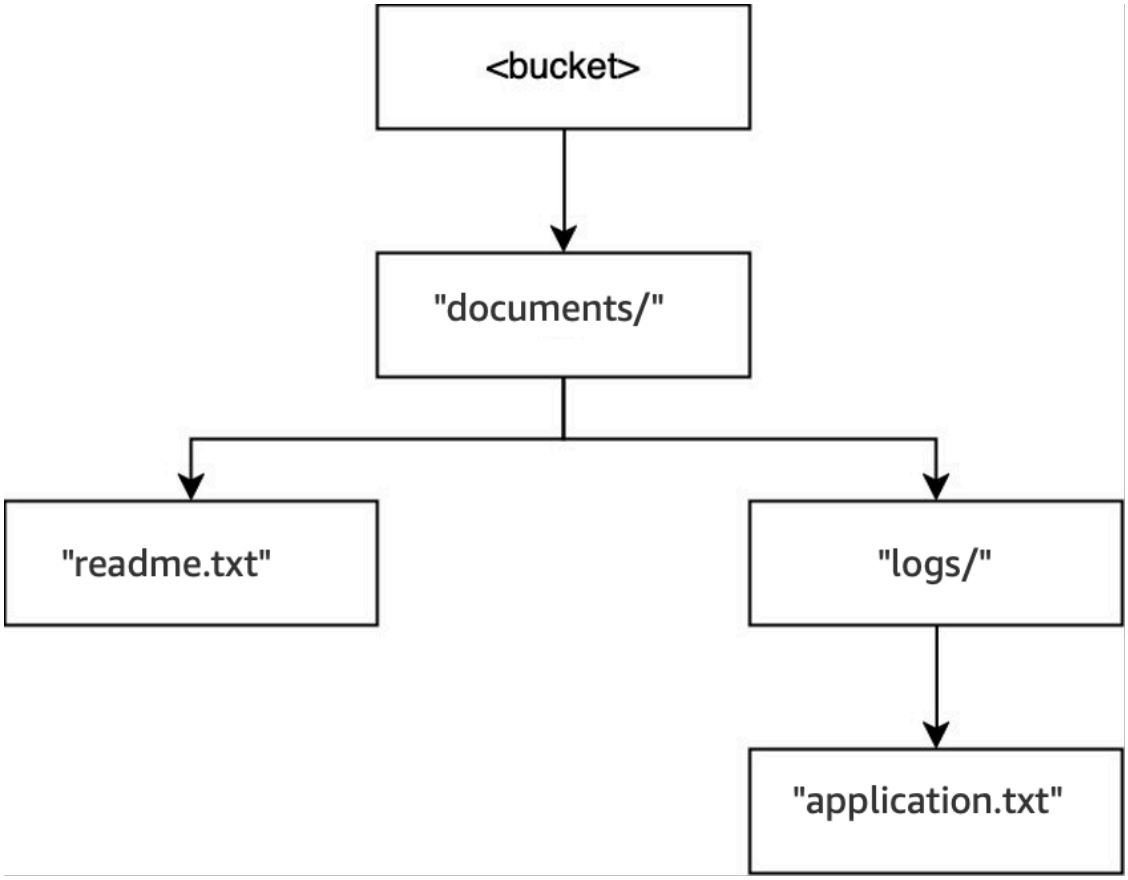
1. When a `LIST(<bucket>)` is executed, the directory `documents/` is entered and the traversing begins.
2. The subdirectory `logs/` is entered and the traversing begins.
3. The object `application.txt` is found within `logs/`.
4. No more entries exist within `logs/`. The List operation exits from `logs/` and enters `documents/` again.
5. The `documents/` directory continues being traversed and the object `readme.txt` is found.
6. The `documents/` directory continues being traversed and the subdirectory `reports/` is entered and the traversing begins.
7. The object `quarterly.txt` is found within `reports/`.
8. No more entries exist within `reports/`. The List exits from `reports/` and enters `documents/` again.
9. No more entries exist within `documents/` and the List returns.

In this example, `logs/` is ordered before `readme.txt` and `readme.txt` is ordered before `reports/`.

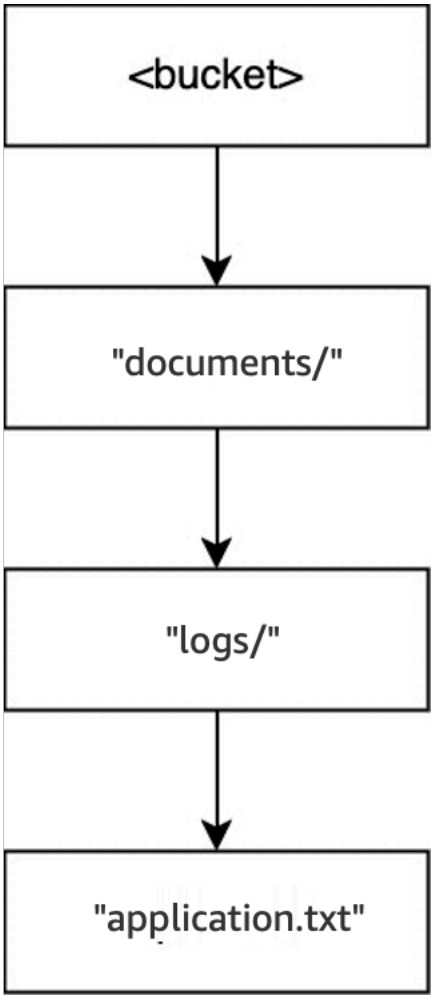
Example 3: How S3 `DeleteObject` requests to a directory bucket interact with directories



1. In that same bucket, when the operation `DELETE(<bucket>, "documents/reports/quarterly.txt")` is executed, the object `quarterly.txt` is deleted, leaving the directory `reports/` empty and causing it to be deleted immediately. The `documents/` directory is not empty because it has both the directory `logs/` and the object `readme.txt` within it, so it's not deleted. For this operation, only one object and one directory were deleted.



2. When the operation `DELETE(<bucket>, "documents/readme.txt")` is executed, the object `readme.txt` is deleted. `documents/` is still not empty because it contains the directory `logs/`, so it's not deleted. For this operation, no directories are deleted and only the object is deleted.



3. Lastly, when the operation `DELETE(<bucket>, "documents/logs/application.txt")` is executed, `application.txt` is deleted, leaving `logs/` empty and causing it to be deleted immediately. This then leaves `documents/` empty and causing it to also be deleted immediately. For this operation, two directories and one object are deleted. The bucket is now empty.



Data residency workloads

S3 directory buckets are available in AWS Dedicated Local Zones and in select AWS Local Zones. AWS Dedicated Local Zones (Dedicated Local Zones) are a type of AWS infrastructure that is fully managed by AWS, built for exclusive use by you or your community, and placed in a location or

data center specified by you to help comply with regulatory requirements. Dedicated Local Zones are a type of AWS Local Zones (Local Zones) offering. For more information, see AWS Dedicated Local Zones.

In Dedicated Local Zones, you can create S3 directory buckets to store data in a specific data perimeter, which helps support data residency and isolation use cases. Directory buckets in Dedicated Local Zones can support the S3 Express One Zone and S3 One Zone-Infrequent Access (S3 One Zone-IA; Z-IA) storage classes.

A Local Zone is a type of AWS infrastructure that places select compute, storage, and other services near large populations and industry centers. To learn more, see AWS Local Zones. You can create S3 directory buckets to store data in a specific data perimeter, which helps support data residency use cases. Directory buckets in Local Zones support the S3 One Zone-Infrequent Access (S3 One Zone-IA; Z-IA) storage class. For more information about supported locations, see AWS Local Zones.

You can use the directory buckets in Local Zones using the AWS Management Console, REST API, AWS Command Line Interface (AWS CLI), and AWS SDKs.

For more information about working with the directory buckets in Local Zones, see the following topics:

Topics

- Concepts for directory buckets in Local Zones
- Enable accounts for Local Zones
- Private connectivity from your VPC
- Creating a directory bucket in a Local Zone
- Authenticating and authorizing for directory buckets in Local Zones

Concepts for directory buckets in Local Zones

Before creating a directory bucket in a Local Zone, you must have the Local Zone ID where you want to create a bucket. You can find all Local Zone information by using the DescribeAvailabilityZones API operation. This API operation lists information about Local Zones, including their Local Zone IDs, parent Region names, network border groups, and opt-in status. After you have your Local Zone ID and you are opted in, you can create a directory bucket in the Local Zone. A directory bucket name consists of a base name that you provide and a suffix that contains the Zone ID of your bucket location, followed by `--x-s3`.

A Local Zone is connected to the **parent Region** using the Amazon redundant and very high-bandwidth private network. This gives applications running in the Local Zone fast, secure, and seamless access to the rest of the AWS services in the parent Region. **Parent Zone ID** is the ID of the zone that handles the Local Zone control plane operations. **Network Border Group** is a unique group from which AWS advertises public IP addresses. For more information about Local Zones, parent Region, and parent Zone ID, see AWS Local Zones concepts in the AWS Local Zones *User Guide*.

All directory buckets use the `s3express` namespace, which is separate from the `s3` namespace for general purpose buckets. For directory buckets, requests are routed to either a **Regional endpoint** or a **Zonal endpoint**. The routing is handled automatically for you if you use the AWS Management Console, AWS CLI, or AWS SDKs.

Most bucket-level API operations (such as `CreateBucket` and `DeleteBucket`) are routed to Regional endpoints, and are referred to as Regional endpoint API operations. Regional endpoints are in the format of `s3express-control.ParentRegionCode.amazonaws.com`. All object-level API operations (such as `PutObject`) and two bucket-level API operations (`CreateSession` and `HeadBucket`) are routed to Zonal endpoints, and are referred to as Zonal endpoint API operations. Zonal endpoints are in the format of `s3express-LocalZoneID.ParentRegionCode.amazonaws.com`. For a complete list of API operations by endpoint type, see Directory bucket API operations.

S3 is available in the Beijing Local Zone:

Local Zone ID: `cnn1-pkx1-az1`

- **Parent Region Name:** `China (Beijing)`
- **Parent Region Code:** `cn-north-1`
- **Parent Zone IDs:** `cnn1-az1`
- **Local Zone Name:** `China (Beijing)`
- **Local Zone Code:** `cnn1-pkx1-az1`
- **Network Border Group:** `cn-north-1-pkx-1`
- **Example bucket ARN:** `arn:aws-cn:s3express:cn-north-1:`*`123456789012`*`:bucket/`*`amzn-s3-demo-bucket`*`--cnn1-pkx1-az1--x-s3`
- **Regional endpoint:** `s3express-control.cn-north-1.amazonaws.com.cn`

Zonal endpoint: `s3express-cnn1-pkx1-az1.cn-north-1.amazonaws.com.cn`

Storage class: `S3 One Zone-Infrequent Access (S3 One Zone-IA; Z-IA)`

To access directory buckets in Local Zones from your virtual private cloud (VPC), you can use gateway VPC endpoints. There is no additional charge for using gateway endpoints. To configure gateway VPC endpoints to access directory buckets and objects in Local Zones, see Private connectivity from your VPC.

Enable accounts for Local Zones

The following topics describe how accounts are enabled for AWS Local Zones:

AWS Local Zones

To get started using AWS Local Zones, you must first opt in to a Local Zone through the AWS Global View console or the AWS CLI. To learn more, see Getting started with AWS Local Zones. You can use the DescribeAvailabilityZones API operation to confirm your account ID access to a list of AWS Local Zones.

AWS Dedicated Local Zones

For all the services in AWS Dedicated Local Zones (Dedicated Local Zones), including Amazon S3, your administrator must enable your AWS account before you can create or access any resource in the Dedicated Local Zone. You can use the DescribeAvailabilityZones API operation to confirm your account ID access to a Local Zone.

Data protection for directory buckets in Local Zones

To further protect your data in Amazon S3, by default, you only have access to the S3 resources that you create. Buckets in Local Zones have all S3 Block Public Access settings enabled by default and S3 Object Ownership is set to bucket owner enforced. These settings can't be modified. Optionally, to restrict access to only within the Local Zone network border groups, you can use the condition key `s3express:AllAccessRestrictedToLocalZoneGroup` in your IAM policies. For more information, see Authenticating and authorizing for directory buckets in Local Zones.

Private connectivity from your VPC

To reduce network latency, configure your virtual private cloud (VPC) with a gateway endpoint to access directory buckets in Availability Zones. This keeps traffic within the AWS network at no additional cost.

To configure a gateway VPC endpoint

1. Open the Amazon VPC Console.
2. In the navigation pane, choose **Endpoints**.
3. Choose **Create endpoint**.
4. Create a name for your endpoint.
5. For **Service category**, choose **AWS services**.
6. For **Services**, add the filter **Type=Gateway** and then choose the option button next to **com.amazonaws.*region*.s3express**.
7. For **VPC**, choose the VPC in which to create the endpoint.
8. For **Route tables**, choose the route table in your VPC to be used by the endpoint. After the endpoint is created, a route record will be added to the route table that you select in this step.
9. For **Policy**, choose **Full access** to allow all operations by all principals on all resources over the VPC endpoint. Otherwise, choose **Custom** to attach a VPC endpoint policy that controls the principals' permissions to perform actions on resources over the VPC endpoint.
10. For **IP address type**, choose from the following options:

    - **IPv4** – Assign IPv4 addresses to the endpoint network interfaces. This option is supported only if all selected subnets have IPv4 address ranges and the service accepts IPv4 requests.
    - **IPv6** – Assign IPv6 addresses to the endpoint network interfaces. This option is supported only if all selected subnets are IPv6 only subnets and the service accepts IPv6 requests.
    - **Dualstack** – Assign both IPv4 and IPv6 addresses to the endpoint network interfaces. This option is supported only if all selected subnets have both IPv4 and IPv6 address ranges and the service accepts both IPv4 and IPv6 requests.

11. (Optional) To add a tag, choose **Add new tag**, and enter the tag key and the tag value.
12. Choose **Create endpoint**.

To learn more about gateway VPC endpoints, see Gateway endpoints in the *AWS PrivateLink Guide*. For data residency use cases, enable access to your buckets only from your VPC using gateway VPC endpoints. When access is restricted to a VPC or a VPC endpoint, you can access the objects through the AWS Management Console, the REST API, AWS CLI, and AWS SDKs.

Note

To restrict access to a VPC or a VPC endpoint using the AWS Management Console, you must use the AWS Management Console Private Access. For more information, see AWS Management Console Private Access in the *AWS Management Console guide*.

Creating a directory bucket in a Local Zone

In Local Zones, you can create directory buckets to store and retrieve objects in a specific data perimeter to help meet your data residency use cases. S3 directory buckets are the only supported bucket type in Local Zones, and contain a bucket location type called `LocalZone`. A directory bucket name consists of a base name that you provide and a suffix that contains the Zone ID of your bucket location and `--x-s3`. You can obtain a list of Local Zone IDs by using the DescribeAvailabilityZones API operation. For more information, see Directory bucket naming rules.

Note

- For all the services in AWS Dedicated Local Zones (Dedicated Local Zones), including S3, your administrator must enable your AWS account before you can create or access any resource in the Dedicated Local Zone. For more information, see Enable accounts for Local Zones.
- For the data residency requirements, we recommend enabling access to your buckets only from gateway VPC endpoints. For more information, see Private connectivity from your VPC.
- To restrict access to only within the Local Zone network border groups, you can use the condition key `s3express:AllAccessRestrictedToLocalZoneGroup` in your IAM policies. For more information, see Authenticating and authorizing for directory buckets in Local Zones.

The following describes ways to create a directory bucket in a single Local Zone with the AWS Management Console, AWS CLI, and AWS SDKs.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the parent Region of a Local Zone in which you want to create a directory bucket.

   > **Note**
   >
   > For more information about the parent Regions, see Concepts for directory buckets in Local Zones.

3. In the left navigation pane, choose **Buckets**.
4. Choose **Create bucket**.

   The **Create bucket** page opens.
5. Under **General configuration**, view the AWS Region where your bucket will be created.
6. Under **Bucket type**, choose **Directory**.

   > **Note**
   >
   > - If you've chosen a Region that doesn't support directory buckets, the bucket type defaults to a general purpose bucket. To create a directory bucket, you must choose a supported Region. For a list of Regions that support directory buckets, see the section called "Regional and Zonal endpoints for directory buckets".
   > - After you create the bucket, you can't change the bucket type.

7. Under **Bucket location**, choose a Local Zone that you want to use.

   > **Note**
   >
   > The Local Zone can't be changed after the bucket is created.

8. Under **Bucket location**, select the checkbox to acknowledge that in the event of a Local Zone outage, your data might be unavailable or lost.

> **Important**
>
> Although directory buckets are stored across multiple devices within a single Local Zone, directory buckets don't store data redundantly across Local Zones.

9. For **Bucket name**, enter a name for your directory bucket.

   For more information about the naming rules for directory buckets, see General purpose bucket naming rules. A suffix is automatically added to the base name that you provide when you create a directory bucket using the console. This suffix includes the Zone ID of the Local Zone that you chose.

   After you create the bucket, you can't change its name.

   > **Important**
   >
   > Don't include sensitive information, such as account numbers, in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

10. Under **Object Ownership**, the **Bucket owner enforced** setting is automatically enabled, and all access control lists (ACLs) are disabled. For directory buckets, ACLs are disabled and can't be enabled.

    With the **Bucket owner enforced** setting enabled, the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect access permissions to data in the S3 bucket. The bucket uses policies exclusively to define access control. A majority of modern use cases in Amazon S3 no longer require the use of ACLs. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

11. Under **Block Public Access settings for this bucket**, all Block Public Access settings for your directory bucket are automatically enabled. These settings can't be modified for directory buckets. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

12. Under **Default encryption**, directory buckets use **Server-side encryption with Amazon S3 managed keys (SSE-S3)** to encrypt data by default. You also have the option to encrypt data in directory buckets with **Server-side encryption with AWS Key Management Service keys (SSE-KMS)**.

13. Choose **Create bucket**.

After creating the bucket, you can add files and folders to the bucket. For more information, see the section called "Working with objects in a directory bucket".

Using the AWS CLI

This example shows how to create a directory bucket in a Local Zone by using the AWS CLI. To use the command, replace the *user input placeholders* with your own information.

When you create a directory bucket, you must provide configuration details and use the following naming convention: *bucket-base-name*--*zone-id*--x-s3.

```
aws s3api create-bucket
--bucket bucket-base-name--zone-id--x-s3
--create-bucket-configuration 'Location={Type=LocalZone,Name=local-zone-
id},Bucket={DataRedundancy=SingleLocalZone,Type=Directory}'
--region parent-region-code
```

For more information about Local Zone ID and Parent Region Code, see Concepts for directory buckets in Local Zones. For more information about the AWS CLI command, see create-bucket in the *AWS CLI Command Reference*.

Using the AWS SDKs

SDK for Go

This example shows how to create a directory bucket in a Local Zone by using the AWS SDK for Go.

Example

```
var bucket = "bucket-base-name--zone-id--x-s3" // The full directory bucket name

func runCreateBucket(c *s3.Client) {
    resp, err := c.CreateBucket(context.Background(), &s3.CreateBucketInput{
        Bucket: &bucket,
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            Location: &types.LocationInfo{
                Name: aws.String("local-zone-id"),
                Type: types.LocationTypeLocalZone,
            },
```

```
            Bucket: &types.BucketInfo{
                DataRedundancy: types.DataRedundancySingleLocalZone,
                Type:           types.BucketTypeDirectory,
            },
        },
    })
    var terr *types.BucketAlreadyOwnedByYou
    if errors.As(err, &terr) {
        fmt.Printf("BucketAlreadyOwnedByYou: %s\n", aws.ToString(terr.Message))
        fmt.Printf("noop...\n") // No operation performed, just printing a message
        return
    }
    if err != nil {
        log.Fatal(err)
    }

    fmt.Printf("bucket created at %s\n", aws.ToString(resp.Location))
}
```

SDK for Java 2.x

This example shows how to create a directory bucket in a Local Zone by using the AWS SDK for Java 2.x.

Example

```
public static void createBucket(S3Client s3Client, String bucketName) {

    //Bucket name format is {base-bucket-name}--{local-zone-id}--x-s3
    //example: doc-example-bucket--local-zone-id--x-s3 is a valid name for a
 directory bucket created in a Local Zone.

    CreateBucketConfiguration bucketConfiguration =
 CreateBucketConfiguration.builder()
             .location(LocationInfo.builder()
                     .type(LocationType.LOCAL_ZONE)
                     .name("local-zone-id").build()) //this must match the Local
 Zone ID in your bucket name
             .bucket(BucketInfo.builder()
                    .type(BucketType.DIRECTORY)
                    .dataRedundancy(DataRedundancy.SINGLE_LOCAL_ZONE)
                    .build()).build();
    try {
```

```
            CreateBucketRequest bucketRequest =
 CreateBucketRequest.builder().bucket(bucketName).createBucketConfiguration(bucketConfigurat
            CreateBucketResponse response = s3Client.createBucket(bucketRequest);
            System.out.println(response);
    }

    catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
         }
    }
```

AWS SDK for JavaScript

This example shows how to create a directory bucket in a Local Zone by using the AWS SDK for JavaScript.

Example

```
// file.mjs, run with Node.js v16 or higher
// To use with the preview build, place this in a folder
// inside the preview build directory, such as /aws-sdk-js-v3/workspace/

import { S3 } from "@aws-sdk/client-s3";

const region = "parent-region-code";
const zone = "local-zone-id";
const suffix = `${zone}--x-s3`;

const s3 = new S3({ region });

const bucketName = `bucket-base-name--${suffix}`; // Full directory bucket name

const createResponse = await s3.createBucket(
    { Bucket: bucketName,
      CreateBucketConfiguration: {Location: {Type: "LocalZone", Name: "local-zone-
id"},
      Bucket: { Type: "Directory", DataRedundancy: "SingleLocalZone" }}
    }
   );
```

SDK for .NET

This example shows how to create a directory bucket in a Local Zone by using the SDK for .NET.

Example

```
using (var amazonS3Client = new AmazonS3Client())
{
    var putBucketResponse = await amazonS3Client.PutBucketAsync(new PutBucketRequest
    {

       BucketName = "bucket-base-name--local-zone-id--x-s3",
       PutBucketConfiguration = new PutBucketConfiguration
       {
         BucketInfo = new BucketInfo { DataRedundancy =
 DataRedundancy.SingleLocalZone, Type = BucketType.Directory },
         Location = new LocationInfo { Name = "local-zone-id", Type =
 LocationType.LocalZone }
       }
     }).ConfigureAwait(false);
}
```

SDK for PHP

This example shows how to create a directory bucket in a Local Zone by using the AWS SDK for PHP.

Example

```
require 'vendor/autoload.php';

$s3Client = new S3Client([

    'region'      => 'parent-region-code',
]);


$result = $s3Client->createBucket([
    'Bucket' => 'bucket-base-name--local-zone-id--x-s3',
    'CreateBucketConfiguration' => [
        'Location' => ['Name'=> 'local-zone-id', 'Type'=> 'LocalZone'],
        'Bucket' => ["DataRedundancy" => "SingleLocalZone" ,"Type" => "Directory"]
  ],
```

```
]);
```

SDK for Python (Boto3)

This example shows how to create a directory bucket in a Local Zone by using the AWS SDK for Python (Boto3).

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def create_bucket(s3_client, bucket_name, local_zone):
    '''
    Create a directory bucket in a specified Local Zone

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to create; for example, 'bucket-base-name--local-
zone-id--x-s3'
    :param local_zone: String; Local Zone ID to create the bucket in
    :return: True if bucket is created, else False
    '''

    try:
        bucket_config = {
                'Location': {
                    'Type': 'LocalZone',
                    'Name': local_zone
                },
                'Bucket': {
                    'Type': 'Directory',
                    'DataRedundancy': 'SingleLocalZone'
                }
            }
        s3_client.create_bucket(
            Bucket = bucket_name,
            CreateBucketConfiguration = bucket_config
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True
```

```
if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'parent-region-code'
    local_zone = 'local-zone-id'
    s3_client = boto3.client('s3', region_name = region)
    create_bucket(s3_client, bucket_name, local_zone)
```

SDK for Ruby

This example shows how to create an directory bucket in a Local Zone by using the AWS SDK for Ruby.

Example

```
s3 = Aws::S3::Client.new(region:'parent-region-code')
s3.create_bucket(
  bucket: "bucket-base-name--local-zone-id--x-s3",
  create_bucket_configuration: {
    location: { name: 'local-zone-id', type: 'LocalZone' },
    bucket: { data_redundancy: 'SingleLocalZone', type: 'Directory' }
  }
)
```

Authenticating and authorizing for directory buckets in Local Zones

Directory buckets in Local Zones support both AWS Identity and Access Management (IAM) authorization and session-based authorization. For more information about authentication and authorization for directory buckets, see Authenticating and authorizing requests.

Resources

Amazon Resource Names (ARNs) for directory buckets contain the `s3express` namespace, the AWS parent Region, the AWS account ID, and the directory bucket name which includes the Zone ID. To access and perform actions on your directory bucket, you must use the following ARN format:

```
arn:aws:s3express:region-code:account-id:bucket/bucket-base-name--ZoneID--x-s3
```

For directory buckets in a Local Zone, the Zone ID is the ID of the Local Zone. For more information about directory buckets in Local Zones, see Concepts for directory buckets in Local Zones. For more information about ARNs, see Amazon Resource Names (ARNs) in the *IAM User Guide*. For more information about resources, see IAM JSON Policy Elements: Resource in the *IAM User Guide*.

Condition keys for directory buckets in Local Zones

In Local Zones, you can use all of these condition keys in your IAM policies. Additionally, to create a data perimeter around your Local Zone network border groups, you can use the condition key `s3express:AllAccessRestrictedToLocalZoneGroup` to deny all requests from outside the groups.

The following condition key can be used to further refine the conditions under which an IAM policy statement applies. For a complete list of API operations, policy actions, and condition keys that are supported by directory buckets, see Policy actions for directory buckets.

> **Note**
>
> The following condition key only applies to Local Zones and isn't supported in Availability Zones and AWS Regions.

API operations	Policy actions	Description	Condition key	Description	Type
Zonal endpoint API operations	`s3express:CreateSession`	Grants permission to create a session token, which is used for granting access to all Zonal endpoint API operations, such as `CreateSession` , `HeadBucket` , `CopyObject` , `PutObject` , and `GetObject` .	`s3express:AllAccessRestrictedToLocalZoneGroup`	Filters all access to the bucket unless the request originates from the AWS Local Zone network border groups provided in this condition key. **Values:** Local Zone network border group value	`String`

Example policies

To restrict object access to requests from within a data residency boundary that you define (specifically, a Local Zone Group which is a set of Local Zones parented to the same AWS Region), you can set any of the following policies:

- The service control policy (SCP). For information about SCPs, see Service control policies (SCPs) in the *AWS Organizations User Guide*.
- The IAM identity-based policy for the IAM role.
- The VPC endpoint policy. For more information about the VPC endpoint policies, see Control access to VPC endpoints using endpoint policies in the *AWS PrivateLink Guide*.
- The S3 bucket policy.

> **Note**
>
> The condition key `s3express:AllAccessRestrictedToLocalZoneGroup` doesn't support access from an on-premises environment. To support the access from an on-premises environment, you must add the source IP to the policies. For more information, see aws:SourceIp in the IAM User Guide.

Example– SCP policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Access-to-specific-LocalZones-only",
            "Effect": "Deny",
            "Action": [
                "s3express:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEqualsIfExists": {
                    "s3express:AllAccessRestrictedToLocalZoneGroup": [
                        "local-zone-network-border-group-value"
                    ]
```

```
                }
            }
        }
    ]
}
```

Example– IAM identity-based policy (attached to IAM role)

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "s3express:CreateSession",
        "Resource": "*",
        "Condition": {
            "StringNotEqualsIfExists": {
                "s3express:AllAccessRestrictedToLocalZoneGroup": [
                    "local-zone-network-border-group-value"
                ]
            }
        }
    }
}
```

Example– VPC endpoint policy

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Access-to-specific-LocalZones-only",
            "Principal": "*",
```

```
            "Action": "s3express:CreateSession",
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                 "StringNotEqualsIfExists": {
                     "s3express:AllAccessRestrictedToLocalZoneGroup": [
                         "local-zone-network-border-group-value"
                     ]
                 }
            }
        }
    ]
}
```

Example– bucket policy

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Access-to-specific-LocalZones-only",
            "Principal": "*",
            "Action": "s3express:CreateSession",
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                 "StringNotEqualsIfExists": {
                     "s3express:AllAccessRestrictedToLocalZoneGroup": [
                         "local-zone-network-border-group-value"
                     ]
                 }
            }
        }
    ]
}
```

Differences for directory buckets

When using Amazon S3, you can choose the bucket type that best fits your application and performance requirements. A directory bucket is a type of bucket that is best used for low latency or data residency use cases. To learn more about directory buckets, see Working with directory buckets.

For more information about how directory buckets are different, see the following topics.

Topics

- Differences for directory buckets
- API operations supported for directory buckets
- Amazon S3 features not supported by directory buckets

Differences for directory buckets

- **Directory bucket names**
  - A directory bucket name consists of a base name that you provide and a suffix that contains the ID of the AWS Zone (an Availability Zone or Local Zone) that your bucket is located in, followed by `--x-s`. For a list of rules and examples of directory bucket names, see Directory bucket naming rules.
- **`ListObjectsV2` behavior**
  - For directory buckets, `ListObjectsV2` does not return objects in lexicographical (alphabetical) order. Additionally, prefixes must end in a delimiter and only "/" can be specified as the delimiter.
  - For directory buckets, `ListObjectsV2` response includes the prefixes that are related only to in-progress multipart uploads.
- **Deletion behavior** – When you delete an object in a directory bucket, Amazon S3 recursively deletes any empty directories in the object path. For example, if you delete the object key `dir1/dir2/file1.txt`, Amazon S3 deletes `file1.txt`. If the `dir1/` and `dir2/` directories are empty and contain no other objects, Amazon S3 also deletes those directories.
- **ETags and checksums** – Entity tags (ETags) for directory buckets are random alphanumeric strings unique to the object and not MD5 checksums. For more information about using additional checksums with directory buckets, see S3 additional checksum best practices.
- **Object keys in `DeleteObjects` requests**

  - Object keys in `DeleteObjects` requests must contain at least one non-white space character. Strings of all white space characters aren't supported in `DeleteObjects` requests.
  - Object keys in `DeleteObjects` requests cannot contain Unicode control characters, except for the newline (`\n`), tab (`\t`), and carriage return (`\r`) characters.
- **Regional and Zonal endpoints** – Bucket-management API operations for directory buckets are available through a Regional endpoint and are referred to as Regional endpoint API operations. Examples of Regional endpoint API operations are CreateBucket and DeleteBucket. After you create a directory bucket, you can use Zonal endpoint API operations to upload and manage the objects in your directory bucket. Zonal endpoint API operations are available through a Zonal endpoint. Examples of Zonal endpoint API operations are `PutObject` and `CopyObject`. When using directory buckets, you must specify the Region in all requests. For Regional endpoints, you specify the Region, for example, `s3express-control.us-west-2.amazonaws.com`. For Zonal endpoints, you specify both the Region and the Availability Zone, for example, `s3express-usw2-az1.us-west-2.amazonaws.com`. For more information, see Regional and Zonal endpoints for directory buckets.
- **Multipart uploads** – You can upload and copy large objects that are stored in directory buckets by using the multipart upload process. However, the following are some differences when using the multipart upload process with objects stored in directory buckets. For more information, see the section called “Using multipart uploads with directory buckets”.
  - The object creation date is the completion date of the multipart upload.
  - Multipart part numbers must use consecutive part numbers. If you try to complete a multipart upload request with nonconsecutive part numbers, Amazon S3 generates an HTTP `400 (Bad Request)` error.
  - The initiator of a multipart upload can abort the multipart upload request only if they have been granted explicit allow access to `AbortMultipartUpload` through the `s3express:CreateSession` permission. For more information, see Authorizing Regional endpoint API operations with IAM.
- **Emptying a directory bucket** – The `s3 rm` command through the AWS Command Line Interface (CLI), the `delete` operation through Mountpoint, and the **Empty** bucket option button through the AWS Management Console are unable to delete in-progress multipart uploads in a directory bucket. To delete these in-progress multipart uploads, use the `ListMultipartUploads` operation to list the in-progress multipart uploads in the bucket and use the `AbortMultipartUpload` operation to abort all the in-progress multipart uploads.

- **AWS Command Line Interface (CLI) `s3 sync` command** – The `s3 sync` command is not supported with directory buckets. Use `s3 cp --recursive` as an alternative to transfer files to or from directory buckets.
- **AWS Local Zones** – Local Zones are only supported for directory buckets not general purpose buckets.
  - Appending data to existing objects isn't supported for directory buckets that reside in Local Zones. You can only append data to existing objects in directory buckets that reside in Availability Zones.

API operations supported for directory buckets

The directory buckets support both Regional (bucket level, or control plane) and Zonal (object level, or data plane) endpoint API operations. For more information, see Networking for directory buckets and Endpoints and gateway VPC endpoints. For a list of supported API operations see Directory bucket API operations.

Amazon S3 features not supported by directory buckets

The following Amazon S3 features are not supported by directory buckets:

- AWS managed policies
- AWS PrivateLink for S3
- MD5 checksums
- Multi-factor authentication (MFA) delete
- S3 Object Lock
- Requester Pays
- S3 Access Grants
- S3 Event Notifications
- S3 Lifecycle transition actions
- S3 Multi-Region Access Points
- S3 Object Lambda Access Points
- S3 Versioning
- S3 Replication
- Object tags

- S3 Select
- Server access logs
- Static website hosting
- S3 Storage Lens
- S3 Storage Lens groups
- S3 Transfer Acceleration
- Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)
- Server-side encryption with customer-provided keys (SSE-C)
- The option to copy an existing bucket's settings when creating a new bucket in the Amazon S3 console
- Enhanced access denied (HTTP `403 Forbidden`) error messages

Networking for directory buckets

To access directory buckets and objects inside, you use Regional and Zonal API endpoints that are different from the standard Amazon S3 endpoints. Depending on the S3 API operation that you use, either a Zonal or Regional endpoint is required. For a complete list of API operations by endpoint type, see Differences for directory buckets.

You can access both Zonal and Regional API operations through gateway virtual private cloud (VPC) endpoints.

The following topics describe the networking requirements for accessing S3 Express One Zone by using a gateway VPC endpoint.

Topics

- Endpoints
- Configuring VPC gateway endpoints

Endpoints

You can access directory buckets and objects inside from your VPC by using gateway VPC endpoints. Directory buckets use Regional and Zonal API endpoints. Depending on the Amazon S3

API operation that you use, either a Regional or Zonal endpoint is required. There is no additional charge for using gateway endpoints.

Bucket-level (or control plane) API operations are available through Regional endpoints and are referred to as Regional endpoint API operations. Examples of Regional endpoint API operations are `CreateBucket` and `DeleteBucket`. When you create a directory bucket, you choose a single Zone (Availability Zone or Local Zone) where your directory bucket will be created. After you create a directory bucket, you can use Zonal endpoint API operations to upload and manage the objects in your directory bucket.

Object-level (or data plane) API operations are available through Zonal endpoints and are referred to as Zonal endpoint API operations. Examples of Zonal endpoint API operations are `CreateSession` and `PutObject`.

For more information about the endpoints and the locations that support directory buckets in Availability Zones, see Endpoints for directory buckets in Availability Zones.

For more information about the endpoints and the locations that support directory buckets in Local Zones, see Enable accounts for Local Zones.

Configuring VPC gateway endpoints

To configure gateway VPC endpoints for access directory buckets in Availability Zones, see the section called "Configuring VPC gateway endpoints".

To configure gateway VPC endpoints for access directory buckets in Local Zones, see Private connectivity from your VPC.

Directory bucket naming rules

When you create a directory bucket in Amazon S3, the following bucket naming rules apply. For general purpose bucket naming rules, see General purpose bucket naming rules.

A directory bucket name consists of a base name that you provide, and a suffix that contains the ID of the AWS Zone (an Availability Zone or a Local Zone) that your bucket is located in and `--x-s3`. The *`zone-id`* can be the ID of an Availability Zone or a Local Zone.

```
base-name--zoneid--x-s3
```

For example, the following directory bucket name contains the Availability Zone ID `usw2-az1`:

```
bucket-base-name--usw2-az1--x-s3
```

> **Note**
>
> When you create a directory bucket by using the console, a suffix is automatically added to the base name that you provide. This suffix includes the Zone ID of the Zone (Availability Zone or Local Zone) that you chose.
> When you create a directory bucket by using an API, you must provide the full suffix, including the Zone ID, in your request. For a list of Zone IDs, see Endpoints.

The following naming rules apply for directory buckets.

- Be unique within the chosen Zone (AWS Availability Zone or AWS Local Zone).
- Name must be between 3 (min) and 63 (max) characters long, including the suffix.
- Consists only of lowercase letters, numbers and hyphens (-).
- Begin and end with a letter or number.
- Must include the following suffix: `--zone-id--x-s3`.
- Bucket names must not start with the prefix `xn--`.
- Bucket names must not start with the prefix `sthree-`.
- Bucket names must not start with the prefix `sthree-configurator`.
- Bucket names must not start with the prefix `amzn-s3-demo-`.
- Bucket names must not end with the suffix `-s3alias`. This suffix is reserved for access point alias names. For more information, see Access point aliases.
- Bucket names must not end with the suffix `--ol-s3`. This suffix is reserved for Object Lambda Access Point alias names. For more information, see How to use a bucket-style alias for your S3 bucket Object Lambda Access Point.
- Bucket names must not end with the suffix `.mrap`. This suffix is reserved for Multi-Region Access Point names. For more information, see Rules for naming Amazon S3 Multi-Region Access Points.

Viewing directory bucket properties

You can view and configure the properties for an Amazon S3 directory bucket by using the Amazon S3 console. For more information, see Working with directory buckets.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Directory buckets**.
3. In the **Directory buckets** list, choose the name of the bucket that you want to view the properties for.
4. Choose the **Properties** tab.
5. On the **Properties** tab, you can view the following properties for the bucket:

   - **Directory bucket overview** – You can see the AWS Region, Zone (Availability Zone or Local Zone), Amazon Resource Name (ARN), and creation date for the bucket.
   - **Server-side encryption settings** – Amazon S3 applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for all S3 buckets. Amazon S3 encrypts an object before saving it to a disk and decrypts the object when you download it. For more information, see Setting and monitoring default encryption for directory buckets.

     For more information about supported features for directory buckets, see Creating and using directory buckets.

Managing directory bucket policies

You can add, delete, update, and view bucket policies for Amazon S3 directory buckets by using the Amazon S3 console, the AWS SDKs and the AWS CLI. For more information, see the following topics. For more information about supported AWS Identity and Access Management (IAM) actions, see Authorizing Regional endpoint API operations with IAM. For example bucket policies for directory buckets, see Example bucket policies for directory buckets.

Topics

- Adding a bucket policy
- Viewing a bucket policy
- Deleting a bucket policy

Adding a bucket policy

To add a bucket policy to a directory bucket, you can use the Amazon S3 console, the AWS SDKs, or the AWS CLI.

Using the S3 console

To create or edit a bucket policy

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Directory buckets**.
3. In the **Directory buckets** list, choose the name of the bucket that you want to add a policy to.
4. Choose the **Permissions** tab.
5. Under **Bucket policy**, choose **Edit**. The **Edit bucket policy** page appears.
6. To generate a policy automatically, choose **Policy generator**.

   If you choose **Policy generator**, the AWS Policy Generator opens in a new window.

   If you don't want to use the AWS Policy Generator, you can add or edit JSON statements in the **Policy** section.

   a. On the **AWS Policy Generator** page, for **Select Type of Policy**, choose **S3 Bucket Policy**.

   b. Add a statement by entering the information in the provided fields, and then choose **Add Statement**. Repeat this step for as many statements as you want to add. For more information about these fields, see the IAM JSON policy elements reference in the *IAM User Guide*.

   > **Note**
   >
   > For your convenience, the **Edit bucket policy** page displays the **Bucket ARN** (Amazon Resource Name) of the current bucket above the **Policy** text field. You can copy this ARN for use in the statements on the **AWS Policy Generator** page.

   c. After you finish adding statements, choose **Generate Policy**.

   d. Copy the generated policy text, choose **Close**, and return to the **Edit bucket policy** page in the Amazon S3 console.

7. In the **Policy** box, edit the existing policy or paste the bucket policy from the AWS Policy Generator. Make sure to resolve security warnings, errors, general warnings, and suggestions before you save your policy.

   > **Note**
   >
   > Bucket policies are limited to 20 KB in size.

8. Choose **Save changes**, which returns you to the **Permissions** tab.

Using the AWS SDKs

SDK for Java 2.x

Example

PutBucketPolicy AWS SDK for Java 2.x

```
public static void setBucketPolicy(S3Client s3Client, String bucketName, String
 policyText) {

      //sample policy text
      /**
       * policy_statement = {
        *         'Version': '2012-10-17',
        *         'Statement': [
        *             {
        *                 'Sid': 'AdminPolicy',
        *                 'Effect': 'Allow',
        *                 'Principal': {
        *                     "AWS": "111122223333"
        *                 },
        *                 'Action': 's3express:*',
        *                 'Resource':
 'arn:aws:s3express:region:111122223333:bucket/bucket-base-name--zone-id--x-s3'
        *             }
        *         ]
        *     }
        */
        System.out.println("Setting policy:");
        System.out.println("----");
        System.out.println(policyText);
```

```
        System.out.println("----");
        System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

        try {
            PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
                    .bucket(bucketName)
                    .policy(policyText)
                    .build();
            s3Client.putBucketPolicy(policyReq);
            System.out.println("Done!");
        }

        catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

Using the AWS CLI

This example shows how to add a bucket policy to a directory bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

```
aws s3api put-bucket-policy --bucket bucket-base-name--zone-id--x-s3 --policy file://
bucket_policy.json
```

bucket_policy.json:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AdminPolicy",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express*",
```

```
            "Resource": "arn:aws:s3express:us-west-2:111122223333:bucket/amzn-s3-
demo-bucket--usw2-az1--x-s3"
        }
    ]
}
```

For more information, see put-bucket-policy in the AWS Command Line Interface.

Viewing a bucket policy

To view a bucket policy for a directory bucket, use the following examples.

Using the AWS CLI

This example shows how to view the bucket policy attached to a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api get-bucket-policy --bucket bucket-base-name--zone-id--x-s3
```

For more information, see get-bucket-policy in the AWS Command Line Interface.

Deleting a bucket policy

To delete a bucket policy for a directory bucket, use the following examples.

Using the AWS SDKs

SDK for Java 2.x

Example

`DeleteBucketPolicy` AWS SDK for Java 2.x

```
public static void deleteBucketPolicy(S3Client s3Client, String bucketName) {
      try {
          DeleteBucketPolicyRequest deleteBucketPolicyRequest =
 DeleteBucketPolicyRequest
                  .builder()
                  .bucket(bucketName)
```

```
                .build()
        s3Client.deleteBucketPolicy(deleteBucketPolicyRequest);
        System.out.println("Successfully deleted bucket policy");
    }

    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
```

Using the AWS CLI

This example shows how to delete a bucket policy for a directory bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

```
aws s3api delete-bucket-policy --bucket bucket-base-name--zone-id--x-s3
```

For more information, see delete-bucket-policy in the AWS Command Line Interface.

Emptying a directory bucket

You can empty an Amazon S3 directory bucket by using the Amazon S3 console. For more information about directory buckets, see Working with directory buckets.

Before you empty a directory bucket, note the following:

- When you empty a directory bucket, you delete all the objects, but you keep the directory bucket.
- After you empty a directory bucket, the empty action can't be undone.
- Objects that are added to the directory bucket while the empty bucket action is in progress might be deleted.

If you also want to delete the bucket, note the following:

- All objects in the directory bucket must be deleted before the bucket itself can be deleted.
- In-progress multipart uploads in the directory bucket must be aborted before the bucket itself can be deleted.

> **Note**
>
> The `s3 rm` command through the AWS Command Line Interface (CLI), the `delete` operation through Mountpoint, and the **Empty** bucket option button through the AWS Management Console are unable to delete in-progress multipart uploads in a directory bucket. To delete these in-progress multipart uploads, use the `ListMultipartUploads` operation to list the in-progress multipart uploads in the bucket and use the `AbortMultipartUpload` operation to abort all the in-progress multipart uploads.

To delete a directory bucket, see Deleting a directory bucket. To abort an in-progress multipart upload, see the section called "Aborting a multipart upload".

To empty a general purpose bucket, see Emptying a general purpose bucket.

Using the S3 console

To empty a directory bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Directory buckets**.
3. Choose the option button next to the name of the bucket that you want to empty, and then choose **Empty**.
4. On the **Empty bucket** page, confirm that you want to empty the bucket by entering **`permanently delete`** in the text field, and then choose **Empty**.
5. Monitor the progress of the bucket emptying process on the **Empty bucket: status** page.

Deleting a directory bucket

You can delete only empty Amazon S3 directory buckets. Before you delete your directory bucket, you must delete all objects in the bucket and abort all in-progress multipart uploads.

If the directory bucket is attached to an access point, you must delete the access point first. For more information, see Delete your access point for directory buckets.

To empty a directory bucket, see Emptying a directory bucket. To abort an in-progress multipart upload, see the section called "Aborting a multipart upload".

To delete a general purpose bucket, see Deleting a general purpose bucket.

Using the S3 console

After you empty your directory bucket and abort all in-progress multipart uploads, you can delete your bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Directory buckets**.
3. In the **Directory buckets** list, choose the option button next to the bucket that you want to delete.
4. Choose **Delete**.
5. On the **Delete bucket** page, enter the name of the bucket in the text field to confirm the deletion of your bucket.

   > **Important**
   >
   > Deleting a directory bucket can't be undone.

6. To delete your directory bucket, choose **Delete bucket**.

Using the AWS SDKs

The following examples delete a directory bucket by using the AWS SDK for Java 2.x and AWS SDK for Python (Boto3).

SDK for Java 2.x

Example

```
public static void deleteBucket(S3Client s3Client, String bucketName) {

    try {
        DeleteBucketRequest del = DeleteBucketRequest.builder()
                .bucket(bucketName)
```

```
                .build();
        s3Client.deleteBucket(del);
        System.out.println("Bucket " + bucketName + " has been deleted");
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for Python (Boto3)

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def delete_bucket(s3_client, bucket_name):
    '''
    Delete a directory bucket in a specified Region

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to delete; for example, 'doc-example-bucket--usw2-
az1--x-s3'
    :return: True if bucket is deleted, else False
    '''

    try:
        s3_client.delete_bucket(Bucket = bucket_name)
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    s3_client = boto3.client('s3', region_name = region)
```

Using the AWS CLI

This example shows how to delete a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api delete-bucket --bucket bucket-base-name--zone-id--x-s3 --region us-west-2
```

For more information, see delete-bucket in the AWS Command Line Interface.

Listing directory buckets

The following examples show how to list directory buckets by using the AWS Management Console, AWS SDKs, and AWS CLI.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to view a list of your directory buckets.
3. In the left navigation pane, choose **Directory buckets**. A list of directory buckets appears. To view the objects in the bucket, bucket properties, bucket permissions, metrics, access points associated with the bucket, or to manage the bucket, choose the bucket name.

Using the AWS SDKs

SDK for Java 2.x

Example

The following example lists directory buckets by using the AWS SDK for Java 2.x.

```
public static void listBuckets(S3Client s3Client) {
       try {
           ListDirectoryBucketsRequest listDirectoryBucketsRequest =
ListDirectoryBucketsRequest.builder().build();
```

```
            ListDirectoryBucketsResponse response =
 s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);
            if (response.hasBuckets()) {
                for (Bucket bucket: response.buckets()) {
                    System.out.println(bucket.name());
                    System.out.println(bucket.creationDate());
                }
            }
        }

        catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

SDK for Python (Boto3)

Example

The following example lists directory buckets by using the AWS SDK for Python (Boto3).

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_directory_buckets(s3_client):
'''
Prints a list of all directory buckets in a Region

:param s3_client: boto3 S3 client
:return: True if there are buckets in the Region, else False
'''
try:
    response = s3_client.list_directory_buckets()
    for bucket in response['Buckets']:
        print (bucket['Name'])
except ClientError as e:
    logging.error(e)
    return False
return True
```

```
if __name__ == '__main__':
    region = 'us-east-1'
    s3_client = boto3.client('s3', region_name = region)
    list_directory_buckets(s3_client)
```

SDK for .NET

Example

The following example lists directory buckets by using the AWS SDK for .NET.

```
var listDirectoryBuckets = await amazonS3Client.ListDirectoryBucketsAsync(new
 ListDirectoryBucketsRequest
{
  MaxDirectoryBuckets = 10
  }).ConfigureAwait(false);
```

SDK for PHP

Example

The following example lists directory buckets by using the AWS SDK for PHP.

```
require 'vendor/autoload.php';

$s3Client = new S3Client([
    'region'      => 'us-east-1',
]);
$result = $s3Client->listDirectoryBuckets();
```

SDK for Ruby

Example

The following example lists directory buckets by using the AWS SDK for Ruby.

```
s3 = Aws::S3::Client.new(region:'us-west-1')
s3.list_directory_buckets
```

Using the AWS CLI

The following `list-directory-buckets` example command shows how you can use the AWS CLI to list your directory buckets in the *us-east-1* region. To run this command, replace the *user input placeholders* with your own information.

```
aws s3api list-directory-buckets --region us-east-1
```

For more information, see list-directory-buckets in the *AWS CLI Command Reference*.

Determining whether you can access a directory bucket

The following AWS SDK examples show how to use the `HeadBucket` API operation to determine if an Amazon S3 directory bucket exists and if you have permission to access it.

Using the AWS SDKs

The following AWS SDK for Java 2.x example shows how to determine if a bucket exists and if you have permission to access it.

SDK for Java 2.x

Example

AWS SDK for Java 2.x

```
public static void headBucket(S3Client s3Client, String bucketName) {
   try {
        HeadBucketRequest headBucketRequest = HeadBucketRequest
                .builder()
                .bucket(bucketName)
                .build();
        s3Client.headBucket(headBucketRequest);
        System.out.format("Amazon S3 bucket: \"%s\" found.", bucketName);
   }
```

```
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Using the AWS CLI

The following `head-bucket` example command shows how you can use the AWS CLI to determine if a directory bucket exists and if you have permission to access it. To run this command, replace the user input placeholders with your own information.

```
aws s3api head-bucket --bucket bucket-base-name--zone-id--x-s3
```

For more information, see head-bucket in the *AWS CLI Command Reference*.

Working with objects in a directory bucket

After you create an Amazon S3 directory bucket, you can work with objects by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and the AWS SDKs.

For more information about performing bulk operations, importing, uploading, copying, deleting, and downloading objects in directory buckets, see the following topics.

Topics

- Importing objects into a directory bucket
- Working with S3 Lifecycle for directory buckets
- Using Batch Operations with directory buckets
- Appending data to objects in directory buckets
- Renaming objects in directory buckets
- Uploading objects to a directory bucket
- Copying objects from or to a directory bucket
- Deleting objects from a directory bucket

- Downloading an object from a directory bucket
- Generating presigned URLs to share objects directory bucket
- Retrieving object metadata from directory buckets
- Listing objects from a directory bucket

Importing objects into a directory bucket

After you create a directory bucket in Amazon S3, you can populate the new bucket with data by using the import action. Import is a streamlined method for creating S3 Batch Operations jobs to copy objects from general purpose buckets to directory buckets.

> **Note**
>
> The following limitations apply to import jobs:
>
> - The source bucket and the destination bucket must be in the same AWS Region and account.
> - The source bucket cannot be a directory bucket.
> - Objects larger than 5GB are not supported and will be omitted from the copy operation.
> - Objects in the Glacier Flexible Retrieval, Glacier Deep Archive, Intelligent-Tiering Archive Access tier, and Intelligent-Tiering Deep Archive tier storage classes must be restored before they can be imported.
> - Imported objects with MD5 checksum algorithms are converted to use CRC32 checksums.
> - Imported objects use the Express One Zone storage class, which has a different pricing structure than the storage classes used by general purpose buckets. Consider this difference in cost when importing large numbers of objects.
> - Objects that have annotations are not supported.

When you configure an import job, you specify the source bucket or prefix where the existing objects will be copied from. You also provide an AWS Identity and Access Management (IAM) role that has permissions to access the source objects. Amazon S3 then starts a Batch Operations job that copies the objects and automatically applies appropriate storage class and checksum settings.

To configure import jobs, you use the Amazon S3 console.

Using the Amazon S3 console

To import objects into a directory bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**, and then choose the **Directory** buckets tab. Choose the option button next to the directory bucket that you want to import objects into.
3. Choose **Import**.
4. For **Source**, enter the general purpose bucket (or bucket path including prefix) that contains the objects that you want to import. To choose an existing general purpose bucket from a list, choose **Browse S3**.
5. For **Permission to access and copy source objects**, do one of the following to specify an IAM role with the permissions necessary to import your source objects:
   - To allow Amazon S3 to create a new IAM role on your behalf, choose **Create new IAM role**.
   - To choose an existing IAM role from a list, choose **Choose from existing IAM roles**.
   - To specify an existing IAM role by entering its Amazon Resource Name (ARN), choose **Enter IAM role ARN**, then enter the ARN in the corresponding field.
6. Review the information that's displayed in the **Destination** and **Copied object settings** sections. If the information in the **Destination** section is correct, choose **Import** to start the copy job.

   The Amazon S3 console displays the status of your new job on the **Batch Operations** page. For more information about the job, choose the option button next to the job name, and then on the **Actions** menu, choose **View details**. To open the directory bucket that the objects will be imported into, choose **View import destination**.

Working with S3 Lifecycle for directory buckets

S3 Lifecycle helps you store objects in S3 Express One Zone in directory buckets cost effectively by deleting expired objects on your behalf. To manage the lifecycle of your objects, create an S3 Lifecycle configuration for your directory bucket. An S3 Lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. You can set an Amazon S3 Lifecycle configuration on a directory bucket by using the AWS Command Line Interface (AWS CLI), the AWS SDKs, the Amazon S3 REST API and AWS CloudFormation.

In your lifecycle configuration, you use rules to define actions that you want Amazon S3 to take on your objects. For objects stored in directory buckets, you can create lifecycle rules to expire objects as they age. You can also create lifecycle rules to delete incomplete multipart uploads in directory buckets at a daily frequency.

When you add a Lifecycle configuration to a bucket, the configuration rules apply to both existing objects and objects that you add later. For example, if you add a Lifecycle configuration rule today with an expiration action that causes objects with a specific prefix to expire 30 days after creation, S3 will queue for removal any existing objects that are more than 30 days old and that have the specified prefix.

How S3 Lifecycle for directory buckets is different

For objects in directory buckets, you can create lifecycle rules to expire objects and delete incomplete multipart uploads. However, S3 Lifecycle for directory buckets doesn't support transition actions between storage classes. S3 Lifecycle for directory buckets requires you to grant explicit permissions to the S3 Lifecycle service principal.

S3 Lifecycle permissions

S3 Lifecycle uses the `DeleteObject`, `DeleteObjects`, and `AbortMultipartUpload` API operations to expire objects in directory buckets. To call these operations, S3 Lifecycle uses the `CreateSession` API to obtain temporary security credentials. You must grant the `s3express:CreateSession` permission with `ReadWrite` session mode to the S3 Lifecycle service principal (`lifecycle.s3.amazonaws.com`). If you don't grant this permission, or if an active policy explicitly denies delete permissions to the S3 Lifecycle service principal, S3 Lifecycle cannot expire objects in your directory bucket. For more information about `CreateSession`, see CreateSession in the *Amazon S3 API Reference*.

Using a bucket policy to grant permissions to the S3 Lifecycle service principal

The following bucket policy grants the S3 Lifecycle service principal permission to create sessions for performing operations such as `DeleteObject` and `DeleteObjects`. When no session mode is specified in a `CreateSession` request, the session is created with the maximum allowable privilege by the permissions in (attempting `ReadWrite` first, then `ReadOnly` if `ReadWrite` is not permitted). However, `ReadOnly` sessions are insufficient for lifecycle operations that modify or delete objects. Therefore, this example explicitly requires a `ReadWrite` session mode by using the `s3express:SessionMode` condition key.

Example Bucket policy to allow `CreateSession` with an explicit `ReadWrite` session mode for lifecycle operations

```
 {
   "Version":"2008-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal": {
            "Service":"lifecycle.s3.amazonaws.com"
          },
          "Action":"s3express:CreateSession",
          "Condition": {
             "StringEquals": {
                "s3express:SessionMode": "ReadWrite"
              }
           },
           "Resource":"arn:aws:s3express:us-east-2:412345678921:bucket/amzn-s3-demo-
bucket--use2-az2--x-s3"
       }
   ]
}
```

Monitoring lifecycle rules

For objects stored in directory buckets, S3 Lifecycle generates AWS CloudTrail management and data event logs. For more information, see CloudTrail log file examples for directory buckets.

For more information about creating lifecycle configurations and troubleshooting S3 Lifecycle related issues, see the following topics:

Creating and managing a Lifecycle configuration for your directory bucket

You can create a lifecycle configuration for directory buckets by using the AWS Command Line Interface (AWS CLI), AWS SDKs and REST APIs.

Using the AWS CLI

You can use the following AWS CLI commands to manage S3 Lifecycle configurations:

- `put-bucket-lifecycle-configuration`

- `get-bucket-lifecycle-configuration`
- `delete-bucket-lifecycle`

For instructions on setting up the AWS CLI, see Developing with Amazon S3 using the AWS CLI in the *Amazon S3 API Reference*.

The Amazon S3 Lifecycle configuration is an XML file. But when you're using the AWS CLI, you cannot specify the XML format. You must specify the JSON format instead. The following are example XML lifecycle configurations and the equivalent JSON configurations that you can specify in an AWS CLIcommand.

The following AWS CLI example puts a lifecycle configuration policy on a directory bucket. This policy specifies that all objects that have the flagged prefix (myprefix) and are the defined object size expire after 7 days. To use this example, replace each *`user input placeholder`* with your own information.

Save the lifecycle configuration policy to a JSON file. In this example, the file is named lifecycle1.json.

Example

JSON

```
{
    "Rules": [
        {
        "Expiration": {
            "Days": 7
        },
        "ID": "Lifecycle expiration rule",
        "Filter": {
            "And": {
                "Prefix": "myprefix/",
                "ObjectSizeGreaterThan": 500,
                "ObjectSizeLessThan": 64000
            }
        },
        "Status": "Enabled"
    }
    ]
}
```

Submit the JSON file as part of the `put-bucket-lifecycle-configuration` CLI command. To use this command, replace each *`user input placeholder`* with your own information.

```
aws s3api put-bucket-lifecycle-configuration --region us-west-2 --profile default
 --bucket amzn-s3-demo-bucket--usw2-az1--x-s3 --lifecycle-configuration file://lc-
policy.json --checksum-algorithm crc32c
```

Example

XML

```
<LifecycleConfiguration>
    <Rule>
        <ID>Lifecycle expiration rule</ID>
        <Filter>
            <And>
                <Prefix>myprefix/</Prefix>
                <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>64000</ObjectSizeLessThan>
            </And>
        </Filter>
        <Status>Enabled</Status>
        <Expiration>
             <Days>7</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

Using the AWS SDKs

SDK for Java

Example

```
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import
 software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
```

```
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRuleAndOperator;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import
 software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleResponse;
import software.amazon.awssdk.services.s3.model.AbortIncompleteMultipartUpload;

// PUT a Lifecycle policy
LifecycleRuleFilter objectExpirationFilter =
 LifecycleRuleFilter.builder().and(LifecycleRuleAndOperator.builder().prefix("dir1/").object
LifecycleRuleFilter mpuExpirationFilter =
 LifecycleRuleFilter.builder().prefix("dir2/").build();

LifecycleRule objectExpirationRule =
 LifecycleRule.builder().id("lc").filter(objectExpirationFilter).status("Enabled").expiratio
                    .days(10)
                    .build())
                .build();
LifecycleRule mpuExpirationRule = LifecycleRule.builder().id("lc-
mpu").filter(mpuExpirationFilter).status("Enabled").abortIncompleteMultipartUpload(AbortInco
                        .daysAfterInitiation(10)
                        .build())
                .build();

PutBucketLifecycleConfigurationRequest putLifecycleRequest =
 PutBucketLifecycleConfigurationRequest.builder()
                .bucket("amzn-s3-demo-bucket--usw2-az1--x-s3")
                .checksumAlgorithm(ChecksumAlgorithm.CRC32)
                .lifecycleConfiguration(
                        BucketLifecycleConfiguration.builder()
                                .rules(objectExpirationRule, mpuExpirationRule)
                                .build()
                ).build();

PutBucketLifecycleConfigurationResponse resp =
 client.putBucketLifecycleConfiguration(putLifecycleRequest);

// GET the Lifecycle policy
```

```
GetBucketLifecycleConfigurationResponse getResp =
 client.getBucketLifecycleConfiguration(GetBucketLifecycleConfigurationRequest.builder().buc
s3-demo-bucket--usw2-az1--x-s3").build());

// DELETE the Lifecycle policy
DeleteBucketLifecycleResponse delResp =
 client.deleteBucketLifecycle(DeleteBucketLifecycleRequest.builder().bucket("amzn-
s3-demo-bucket--usw2-az1--x-s3").build());
```

SDK for Go

Example

```
package main

import (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
)
// PUT a Lifecycle policy
func putBucketLifecycleConfiguration(client *s3.Client, bucketName string ) error {
    lifecycleConfig := &s3.PutBucketLifecycleConfigurationInput{
        Bucket: aws.String(bucketName),
        LifecycleConfiguration: &types.BucketLifecycleConfiguration{
            Rules: []types.LifecycleRule{
                {
                    ID:     aws.String("lc"),
                    Filter: &types.LifecycleRuleFilter{
                        And: &types.LifecycleRuleAndOperator{
                            Prefix: aws.String("foo/"),
                            ObjectSizeGreaterThan: aws.Int64(1000000),
                            ObjectSizeLessThan:    aws.Int64(100000000),
                            },
                        },

                    Status: types.ExpirationStatusEnabled,
                    Expiration: &types.LifecycleExpiration{
                        Days: aws.Int32(int32(1)),
                    },
```

```
                },
                {
                    ID:     aws.String("abortmpu"),
                    Filter: &types.LifecycleRuleFilter{
                        Prefix: aws.String("bar/"),
                    },
                    Status: types.ExpirationStatusEnabled,
                    AbortIncompleteMultipartUpload:
 &types.AbortIncompleteMultipartUpload{
                        DaysAfterInitiation: aws.Int32(int32(5)),
                    },
                },
            },
        },
    }
    _, err := client.PutBucketLifecycleConfiguration(context.Background(),
 lifecycleConfig)
    return err
}
// Get the Lifecycle policy
func getBucketLifecycleConfiguration(client *s3.Client, bucketName string) error {
    getLifecycleConfig := &s3.GetBucketLifecycleConfigurationInput{
        Bucket: aws.String(bucketName),
    }

    resp, err := client.GetBucketLifecycleConfiguration(context.Background(),
 getLifecycleConfig)
    if err != nil {
        return err
    }
    return nil
}
// Delete the Lifecycle policy
func deleteBucketLifecycleConfiguration(client *s3.Client, bucketName string) error
 {
    deleteLifecycleConfig := &s3.DeleteBucketLifecycleInput{
        Bucket: aws.String(bucketName),
    }
    _, err := client.DeleteBucketLifecycle(context.Background(),
 deleteLifecycleConfig)
    return err
}
func main() {
```

```
    cfg, err := config.LoadDefaultConfig(context.Background(),
 config.WithRegion("us-west-2")) // Specify your region here
    if err != nil {
        log.Fatalf("unable to load SDK config, %v", err)
    }
    s3Client := s3.NewFromConfig(cfg)
    bucketName := "amzn-s3-demo-bucket--usw2-az1--x-s3"
    putBucketLifecycleConfiguration(s3Client, bucketName)
    getBucketLifecycleConfiguration(s3Client, bucketName)
    deleteBucketLifecycleConfiguration(s3Client, bucketName)
    getBucketLifecycleConfiguration(s3Client, bucketName)
}
```

SDK for .NET

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class LifecycleTest
    {
        private const string bucketName = "amzn-s3-demo-bucket--usw2-az1--x-s3";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddUpdateDeleteLifecycleConfigAsync().Wait();
        }

        private static async Task AddUpdateDeleteLifecycleConfigAsync()
        {
            try
            {
                var lifeCycleConfiguration = new LifecycleConfiguration()
```

```
                {
                    Rules = new List <LifecycleRule>
                        {
                            new LifecycleRule
                            {
                                 Id = "delete rule",
                                  Filter = new LifecycleFilter()
                                 {
                                     LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                     {
                                         Prefix = "projectdocs/"
                                     }
                                 },
                                 Status = LifecycleRuleStatus.Enabled,
                                 Expiration = new LifecycleRuleExpiration()
                                 {
                                       Days = 10
                                 }
                            }
                        }
                };

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Retrieve an existing configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

                // Add a new rule.
                lifeCycleConfiguration.Rules.Add(new LifecycleRule
                {
                    Id = "mpu abort rule",
                    Filter = new LifecycleFilter()
                    {
                        LifecycleFilterPredicate = new LifecyclePrefixPredicate()
                        {
                            Prefix = "YearlyDocuments/"
                        }
                    },
                    Expiration = new LifecycleRuleExpiration()
                    {
                        Days = 10
```

```
                    },
                    AbortIncompleteMultipartUpload = new
LifecycleRuleAbortIncompleteMultipartUpload()
                    {
                        DaysAfterInitiation = 10
                    }
                });

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Verify that there are now two rules.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);
                Console.WriteLine("Expected # of rulest=2; found:{0}",
lifeCycleConfiguration.Rules.Count);

                // Delete the configuration.
                await RemoveLifecycleConfigAsync(client);

                // Retrieve a nonexistent configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
LifecycleConfiguration configuration)
        {

            PutLifecycleConfigurationRequest request = new
PutLifecycleConfigurationRequest
            {
                BucketName = bucketName,
```

```
                Configuration = configuration
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }

        static async Task <LifecycleConfiguration>
 RetrieveLifecycleConfigAsync(IAmazonS3 client)
        {
            GetLifecycleConfigurationRequest request = new
 GetLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
        }

        static async Task RemoveLifecycleConfigAsync(IAmazonS3 client)
        {
            DeleteLifecycleConfigurationRequest request = new
 DeleteLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
    }
}
```

SDK for Python (Boto3)

Example

```
import boto3

client = boto3.client("s3", region_name="us-west-2")
bucket_name = 'amzn-s3-demo-bucket--usw2-az1--x-s3'

client.put_bucket_lifecycle_configuration(
    Bucket=bucket_name,
    ChecksumAlgorithm='CRC32',
    LifecycleConfiguration={
        'Rules': [
```

```
            {
                'ID': 'lc',
                'Filter': {
                    'And': {
                        'Prefix': 'foo/',
                        'ObjectSizeGreaterThan': 1000000,
                        'ObjectSizeLessThan': 100000000,
                    }
                },
                'Status': 'Enabled',
                'Expiration': {
                    'Days': 1
                }
            },
            {
                'ID': 'abortmpu',
                'Filter': {
                    'Prefix': 'bar/'
                },
                'Status': 'Enabled',
                'AbortIncompleteMultipartUpload': {
                    'DaysAfterInitiation': 5
                }
            }
        ]
    }
)

result = client.get_bucket_lifecycle_configuration(
    Bucket=bucket_name
)

client.delete_bucket_lifecycle(
    Bucket=bucket_name
)
```

Troubleshooting S3 Lifecycle issues for directory buckets

Topics

- I set up my lifecycle configuration but objects in my directory bucket are not expiring
- How do I monitor the actions taken by my lifecycle rules?

I set up my lifecycle configuration but objects in my directory bucket are not expiring

S3 Lifecycle for directory buckets utilizes public APIs to delete objects in S3 Express One Zone. To use object level public APIs, you must grant permission to `CreateSession` and allow S3 Lifecycle permission to delete your objects. If you have an active policy that denies deletes, this will prevent you from allowing S3 Lifecycle to delete objects on your behalf.

It's important to configure your bucket policies correctly to ensure that the objects that you want to delete are eligible for expiration. You can check your AWS CloudTrail logs for `AccessDenied` Trails for `CreateSession` API invocations in CloudTrail to verify if access has been denied. Checking your CloudTrail logs can assist you in troubleshooting access issues and identifying the root cause of access denied errors. You can then fix your incorrect access controls by updating the relevant policies.

If you confirm that your bucket policies are set correctly and you are still experiencing issues, we recommend that you review the lifecycle rules to ensure that they are applied to the right subset of objects.

How do I monitor the actions taken by my lifecycle rules?

You can use AWS CloudTrail data event logs to monitor actions taken by S3 Lifecycle in directory buckets. For more information, see CloudTrail log file examples.

Using Batch Operations with directory buckets

You can use Amazon S3 Batch Operations to perform operations on objects stored in S3 buckets. To learn more about S3 Batch Operations, see Performing large-scale batch operations on Amazon S3 objects.

The following topics discuss performing batch operations on objects stored in the S3 Express One Zone storage class in directory buckets.

Topics

- Using Batch Operations with directory buckets
- Key differences

Using Batch Operations with directory buckets

You can perform the **Copy** operation and the **Invoke AWS Lambda function** operations on objects that are stored in directory buckets. With **Copy**, you can copy objects between buckets of the

same type (for example, from a directory bucket to a directory bucket). You can also copy between general purpose buckets and directory buckets. With **Invoke AWS Lambda function**, you can use a Lambda function to perform actions on objects in your directory bucket with code that you define.

Copying objects

You can copy between the same bucket type or between directory buckets and general purpose buckets. When you copy to a directory bucket, you must use the correct Amazon Resource Name (ARN) format for this bucket type. The ARN format for a directory bucket is `arn:aws:s3express:region:account-id:bucket/bucket-base-name--x-s3`.

> **Note**
>
> Copying objects across different AWS Regions isn't supported when the source or destination bucket is in an AWS Local Zone. The source and destination buckets must have the same parent AWS Region. The source and destination buckets can be different bucket location types (Availability Zone or Local Zone).

You can also populate your directory bucket with data by using the **Import** action in the S3 console. **Import** is a streamlined method for creating Batch Operations jobs to copy objects from general purpose buckets to directory buckets. For **Import** copy jobs from general purpose buckets to directory buckets, S3 automatically generates a manifest. For more information, see Importing objects to a directory bucket and Specifying a manifest.

Invoking Lambda functions (`LambdaInvoke`)

There are special requirements for using Batch Operations to invoke Lambda functions that act on directory buckets. For example, you must structure your Lambda request by using a v2 JSON invocation schema, and specify InvocationSchemaVersion 2.0 when you create the job. For more information, see Invoke AWS Lambda function.

Key differences

The following is a list of key differences when you're using Batch Operations to perform bulk operations on objects that are stored in directory buckets with the S3 Express One Zone storage class:

- For directory buckets, SSE-S3 and server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) are supported. If you make a `CopyObject` request that specifies to

use server-side encryption with customer-provided keys (SSE-C) on a directory bucket (source or destination), the response returns an HTTP `400 (Bad Request)` error.

We recommend that the bucket's default encryption uses the desired encryption configuration and you don't override the bucket default encryption in your `CreateSession` requests or PUT object requests. Then, new objects are automatically encrypted with the desired encryption settings. For more information about the encryption overriding behaviors in directory buckets and how to encrypt new object copies in a directory bucket with SSE-KMS, see Specifying server-side encryption with AWS KMS for new object uploads.

S3 Bucket Keys aren't supported, when you copy SSE-KMS encrypted objects from general purpose buckets to directory buckets, from directory buckets to general purpose buckets, or between directory buckets, through the Copy operation in Batch Operations. In this case, Amazon S3 makes a call to AWS KMS every time a copy request is made for a KMS-encrypted object. For more information about using SSE-KMS on directory buckets, see Setting and monitoring default encryption for directory buckets and Using server-side encryption with AWS KMS keys (SSE-KMS) in directory buckets.

- Objects in directory buckets can't be tagged. You can only specify an empty tag set. By default, Batch Operations copies tags. If you copy an object that has tags between general purpose buckets and directory buckets, you receive a `501 (Not Implemented)` response.
- S3 Express One Zone offers you the option to choose the checksum algorithm that is used to validate your data during uploads or downloads. You can select one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms: CRC32, CRC32, SHA-1, and SHA-256. MD5-based checksums are not supported with the S3 Express One Zone storage class.
- By default, all Amazon S3 buckets set the S3 Object Ownership setting to bucket owner enforced and access control lists (ACLs) are disabled. For directory buckets, this setting can't be modified. You can copy an object from general purpose buckets to directory buckets. However, you can't overwrite the default ACL when you copy to or from a directory bucket.
- Regardless of how you specify your manifest, the list itself must be stored in a general purpose bucket. Batch Operations can't import existing manifests from (or save generated manifests to) directory buckets. However, objects described within the manifest can be stored in directory buckets.
- You can use S3 Inventory reports from directory buckets as a source for your manifest, as long as the inventory report is stored in a general purpose bucket. You can also create a manifest file for

objects within a directory bucket by using the `ListObjectsV2` API operation to list the objects, and then insert the list in a CSV file.

Granting access

To perform copy jobs, you must have the following permissions:

- To copy objects from one directory bucket to another directory bucket, you must have the `s3express:CreateSession` permission.
- To copy objects from directory buckets to general purpose buckets, you must have the `s3express:CreateSession` permission and the `s3:PutObject` permission to write the object copy to the destination bucket.
- To copy objects from general purpose buckets to directory buckets, you must have the `s3express:CreateSession` permission and the `s3:GetObject` permission to read the source object that is being copied.

  For more information, see CopyObject in the *Amazon Simple Storage Service API Reference*.
- To invoke a Lambda function, you must grant permissions to your resource based on your Lambda function. To determine which permissions are required, check the corresponding API operations.

Appending data to objects in directory buckets

You can add data to the end of existing objects stored in the S3 Express One Zone storage class in directory buckets. We recommend that you use the ability to append data to an object if the data is written continuously over a period of time or if you need to read the object while you are writing to the object. Appending data to objects is common for use-cases such as adding new log entries to log files or adding new video segments to video files as they are transcoded then streamed. By appending data to objects, you can simplify applications that previously combined data in local storage before copying the final object to Amazon S3.

There is no minimum size requirement for the data you can append to an object. However, the maximum size of the data that you can append to an object in a single request is 5GB. This is the same limit as the largest request size when uploading data using any Amazon S3 API.

With each successful append operation, you create a part of the object and each object can have up to 10,000 parts. This means you can append data to an object up to 10,000 times. If an object

is created using S3 multipart upload, each uploaded part is counted towards the total maximum of 10,000 parts. For example, you can append up to 9,000 times to an object created by multipart upload comprising of 1,000 parts.

> **Note**
>
> If you hit the limit of parts, you will receive a TooManyParts error. You can use the `CopyObject` API to reset the count.

If you want to upload parts to an object in parallel and you don't need to read the parts while the parts are being uploaded, we recommend that you use Amazon S3 multipart upload. For more information, see Using multipart upload.

Appending data to objects is only supported for objects in directory buckets that are stored in the S3 Express One Zone storage class. For more information on S3 Express One Zone Zone, see Getting started with S3 Express One Zone.

To get started appending data to objects in your directory buckets, you can use the AWS SDKs, AWS CLI, and the `PutObject` API . When you make a `PutObject` request, you set the `x-amz-write-offset-bytes` header to the size of the object that you are appending to. To use the `PutObject` API operation, you must use the `CreateSession` API to establish temporary security credentials to access the objects in your directory buckets. For more information, see `PutObject` and `CreateSession` in the *Amazon S3 API Reference*.

Each successful append operation is billed as a `PutObject` request. To learn more about pricing, see `Amazon S3 pricing`.

> **Note**
>
> Starting with the 1.12 release, Mountpoint for Amazon S3 supports appending data to objects stored in S3 Express One Zone. To get started, you must opt-in by setting the `--incremental-upload` flag. For more information on Mountpoint, see Working with Mountpoint.

If you use a CRC (Cyclic Redundancy Check) algorithm while uploading the appended data, you can retrieve full object CRC-based checksums using the `HeadObject` or `GetObject` request. If

you use the SHA-1 or SHA-256 algorithm while uploading your appended data, you can retrieve a checksum of the appended parts and verify their integrity using the SHA checksums returned on prior PutObject responses. For more information, see Data protection and encryption.

Appending data to your objects by using the AWS CLI, AWS SDKs and the REST API

You can append data to your objects by using the AWS Command Line Interface (AWS CLI), AWS SDKs and REST API.

Using the AWS CLI

The following `put-object` example command shows how you can use the AWS CLI to append data to an object. To run this command, replace the *`user input placeholders`* with your own information

```
aws s3api put-object --bucket amzn-s3-demo-bucket--azid--x-s3 --key sampleinput/
file001.bin --body bucket-seed/file001.bin --write-offset-bytes size-of-sampleinput/
file001.bin
```

Using the AWS SDKs

SDK for Java

You can use the AWS SDK for Java to append data to your objects.

```
var putObjectRequestBuilder = PutObjectRequest.builder()
                                              .key(key)
                                              .bucket(bucketName)
                                              .writeOffsetBytes(9);
var response = s3Client.putObject(putObjectRequestBuilder.build());
```

SDK for Python (Boto)

```
s3.put_object(Bucket='amzn-s3-demo-bucket--use2-az2--x-s3', Key='2024-11-05-sdk-
test', Body=b'123456789', WriteOffsetBytes=9)
```

Using the REST API

You can send REST requests to append data to an object. For more information, see `PutObject`.

Renaming objects in directory buckets

Using the `RenameObject` operation, you can atomically rename an existing object in a directory bucket that uses the S3 Express One Zone storage class, without any data movement. You can rename an object by specifying the existing object's name as the source and the new name of the object as the destination within the same directory bucket. The `RenameObject` API operation will not succeed on objects that end with the slash (/) delimiter character. For more information, see Naming Amazon S3 objects.

The `RenameObject` operation is typically completed in milliseconds regardless of the size of the object. This capability accelerates applications like log file management, media processing, and data analytics. Additionally, `RenameObject` preserves all object metadata properties, including the storage class, encryption type, creation date, last modified date, and checksum properties.

> **Note**
>
> `RenameObject` is only supported for objects stored in the S3 Express One Zone storage class.

To grant access to the `RenameObject` operation, we recommend that you use the `CreateSession` operation for session-based authorization. Specifically, you grant the `s3express:CreateSession` permission to the directory bucket in a bucket policy or an identity-based policy. Then, you make the `CreateSession` API call on the directory bucket to obtain a session token. With the session token in your request header, you can make API requests to this operation. After the session token expires, you make another `CreateSession` API call to generate a new session token for use. The AWS CLI and AWS SDKs will create and manage your session including refreshing the session token automatically to avoid service interruptions when a session expires. For more information about authorization, see `CreateSession` in the *Amazon S3 API Reference*. To learn more about Zonal endpoint API operations, see Authorizing Zonal endpoint API operations with `CreateSession`.

If you don't want to overwrite an existing object, you can add the `If-None-Match` conditional header with the value `'*'` in the `RenameObject` request. Amazon S3 returns a `412 Precondition Failed` error if the object name already exists. For more information, see `RenameObject` in the *Amazon S3 API Reference*.

`RenameObject` is a Zonal endpoint API operation (object-level or data plane operation) that is logged to AWS CloudTrail. You can use CloudTrail to gather information on the `RenameObject` operation performed on your objects in directory buckets. For more information, see Logging with AWS CloudTrail for directory buckets and CloudTrail log file examples for directory buckets.

S3 Express One Zone is the only storage class that supports `RenameObject`, which is priced the same as `PUT`, `COPY`, `POST`, and `LIST` requests (per 1,000 requests) in S3 Express One Zone. For more information, see Amazon S3 pricing.

Renaming an object

To rename an object in your directory bucket, you can use the Amazon S3 console, AWS CLI, AWS SDKs, the REST API or Mountpoint for Amazon S3 (version 1.19.0 or higher).

Using the S3 console

To rename an object in a directory bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Buckets**, and then choose the **Directory buckets** tab. Navigate to the Amazon S3 directory bucket that contains the object that you want to rename.
3. Select the check box for the object that you want to rename.
4. On the **Actions** menu, choose **Rename object**.
5. In the **New object name** box, enter the new name for the object.

   > **Note**
   >
   > If you specify the same object name as an existing object, the operation will fail and Amazon S3 returns a `412 Precondition Failed` error. The object key name length can't exceed 1,024 bytes. Prefixes included in the object name count toward the total length.

6. Choose **Rename object**. Amazon S3 renames your object.

Using the AWS CLI

The `rename-object` examples show how you can use the AWS CLI to rename an object. To run these commands, replace the *`user input placeholders`* with your own information

The following example shows how to rename an object with a conditional check on the source object's ETag.

```
aws s3api rename-object \
    --bucket amzn-s3-demo-bucket--usw2-az1--x-s3 \
    --key new-file.txt \
    --rename-source original-file.txt \
    --source-if-match "\"a1b7c3d2e5f6\""
```

This command does the following:

- Renames an object from *original-file.txt* to *new-file.txt* in the *amzn-s3-demo-bucket--usw2-az1--x-s3* directory bucket.
- Only performs the rename if the source object's ETag matches "*a1b7c3d4e5f6*".

If the ETag doesn't match, the operation will fail with a `412 Precondition Failed` error.

The following example shows how to rename an object with a conditional check on the new specified object name.

```
aws s3api rename-object \
    --bucket amzn-s3-demo-bucket--usw2-az1--x-s3 \
    --key new-file.txt \
    --rename-source amzn-s3-demo-bucket--usw2-az1--x-s3/original-file.txt \
    --destination-if-none-match "\"e5f3g7h8i9j0\""
```

This command does the following:

- Renames an object from *original-file.txt* to *new-file.txt* in the *amzn-s3-demo-bucket--usw2-az1--x-s3* directory bucket.
- Only performs the rename operation if the object exists and the object's ETag doesn't match "*e5f3g7h8i9j0*".

If an object already exists with the new specified name and the matching ETag, the operation will fail with a `412 Precondition Failed` error.

Using the AWS SDKs

SDK for Java

You can use the AWS SDK for Java to rename your objects. To use these examples, replace the *user input placeholders* with your own information

The following example demonstrates how to create a `RenameObjectRequest` using the AWS SDK for Java

```
String key = "key";
String newKey = "new-key";
String expectedETag = "e5f3g7h8i9j0";
RenameObjectRequest renameRequest = RenameObjectRequest.builder()
    .bucket(amzn-s3-demo-bucket--usw2-az1--x-s3)
    .key(newKey)
    .renameSource(key)
    .destinationIfMatch(e5f3g7h8i9j0)
    .build();
```

This code does the following:

- Create a request to rename an object from "*key*" to "*new-key*" in the *amzn-s3-demo-bucket--usw2-az1--x-s3* directory bucket.
- Includes a condition that the rename will only occur if the object's ETag matches "*e5f3g7h8i9j0*".
- If the ETag doesn't match or the object doesn't exist, the operation will fail.

The following example shows how to create a `RenameObjectRequest` with a none-match condition using the AWS SDK for Java.

```
String key = "key";
String newKey = "new-key";
String noneMatchETag = "e5f3g7h8i9j0";
RenameObjectRequest renameRequest = RenameObjectRequest.builder()
    .bucket(amzn-s3-demo-bucket--usw2-az1--x-s3)
    .key(newKey)
    .renameSource(key)
    .destinationIfNoneMatch(noneMatchETag)
    .build();
```

This code does the following:

- Creates a request to rename an object from "*key*" to "*new-key*" in the *amzn-s3-demo-bucket--usw2-az1--x-s3* directory bucket.
- Includes a condition using `.destinationIfNoneMatch(noneMatchETag)` that ensures the rename will only occur if the destination object's ETag doesn't match "*e5f3g7h8i9j0*".

The operation will fail with a `412 Precondition Failed` error if an object exists with the new specified name and has the specified ETag.

SDK for Python (Boto3)

You can use the AWS SDK for Python (Boto3) to rename your objects. To use these examples, replace the *user input placeholders* with your own information.

The following example demonstrates how to rename an object using the AWS SDK for Python (Boto3).

```
def basic_rename(bucket, source_key, destination_key):
    try:
        s3.rename_object(
            Bucket=amzn-s3-demo-bucket--usw2-az1--x-s3,
            Key=destination_key,
            RenameSource=f"{source_key}"
        )
        print(f"Successfully renamed {source_key} to {destination_key}")
    except ClientError as e:
        print(f"Error renaming object: {e}")
```

This code does the following:

- Renames an object from *source_key* to *destination_key* in the *amzn-s3-demo-bucket--usw2-az1--x-s3* directory bucket.
- Prints a success message if the renaming of your object is successful or prints an error message if it fails.

The following example demonstrate how to rename an object with the `SourceIfMatch` and `DestinationIfNoneMatch` conditions using the AWS SDK for Python (Boto3).

```
def rename_with_conditions(bucket, source_key, destination_key, source_etag,
 dest_etag):
    try:
        s3.rename_object(
            Bucket=amzn-s3-demo-bucket--usw2-az1--x-s3,
            Key=destination_key,
            RenameSource=f"{amzn-s3-demo-bucket--usw2-az1--x-s3}/{source_key}",
            SourceIfMatch=source_ETag,
            DestinationIfNoneMatch=dest_ETag
        )
        print(f"Successfully renamed {source_key} to {destination_key} with
 conditions")
    except ClientError as e:
        print(f"Error renaming object: {e}")
```

This code does the following:

- Performs a conditional rename operation and applies two conditions, `SourceIfMatch` and `DestinationIfNoneMatch`. The combination of these conditions ensures that the object hasn't been modified and that an object doesn't already exist with the new specified name.
- Renames an object from *source_key* to *destination_key* in the *amzn-s3-demo-bucket--usw2-az1--x-s3* directory bucket.
- Prints a success message if the renaming of your object is successful, or prints an error message if it fails or if conditions aren't met.

SDK for Rust

You can use the SDK for Rust to rename your objects. To use these examples, replace the *user input placeholders* with your own information.

The following example demonstrates how to rename an object in the *amzn-s3-demo-bucket--usw2-az1--x-s3* directory bucket using the SDK for Rust.

```
async fn basic_rename_example(client: &Client) -> Result<(), Box<dyn Error>> {
    let response = client
        .rename_object()
        .bucket(" amzn-s3-demo-bucket--usw2-az1--x-s3")
        .key("new-name.txt")  // New name/path for the object
        .rename_source("old-name.txt")  // Original object name/path
        .send()
```

```
        .await?;
    Ok(())
}
```

This code does the following:

- Creates a request to rename an object from "*old-name.tx*" to "*new-name.txt*" in the *amzn-s3-demo-bucket--usw2-az1--x-s3* directory bucket.
- Returns a `Result` type to handle potential errors.

Using the REST API

You can send REST requests to rename an object. For more information, see `RenameObject` in the *Amazon S3 API Reference*.

Using Mountpoint for Amazon S3

Starting with the 1.19.0 version or higher, Mountpoint for Amazon S3 supports renaming objects in S3 Express One Zone. For more information on Mountpoint, see Working with Mountpoint.

Uploading objects to a directory bucket

After you create an Amazon S3 directory bucket, you can upload objects to it. The following examples show how to upload an object to a directory bucket by using the S3 console and the AWS SDKs. For information about bulk object upload operations with S3 Express One Zone, see Object management.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Directory buckets**.
3. Choose the name of the bucket that you want to upload your folders or files to.
4. In the **Objects** list, choose **Upload**.
5. On the **Upload** page, do one of the following:

   - Drag and drop files and folders to the dotted upload area.
   - Choose **Add files** or **Add folder**, choose the files or folders to upload, and then choose **Open** or **Upload**.

6. Under **Checksums**, choose the **Checksum function** that you want to use.

   (Optional) If you're uploading a single object that's less than 16 MB in size, you can also specify a precalculated checksum value. When you provide a precalculated value, Amazon S3 compares it with the value that it calculates by using the selected checksum function. If the values don't match, the upload won't start.

7. The options in the **Permissions** and **Properties** sections are automatically set to default settings and can't be modified. Block Public Access is automatically enabled, and S3 Versioning and S3 Object Lock can't be enabled for directory buckets.

   (Optional) If you want to add metadata in key-value pairs to your objects, expand the **Properties** section, and then in the **Metadata** section, choose **Add metadata**.

8. To upload the listed files and folders, choose **Upload**.

   Amazon S3 uploads your objects and folders. When the upload is finished, you see a success message on the **Upload: status** page.

Using the AWS SDKs

SDK for Java 2.x

Example

```
public static void putObject(S3Client s3Client, String bucketName, String objectKey,
 Path filePath) {
       //Using File Path to avoid loading the whole file into memory
       try {
           PutObjectRequest putObj = PutObjectRequest.builder()
                   .bucket(bucketName)
                   .key(objectKey)
                   //.metadata(metadata)
                   .build();
           s3Client.putObject(putObj, filePath);
           System.out.println("Successfully placed " + objectKey +" into bucket
 "+bucketName);

       }

       catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
```

```
        }
}
```

SDK for Python (Boto3)

Example

```
import boto3
import botocore
from botocore.exceptions import ClientError


def put_object(s3_client, bucket_name, key_name, object_bytes):
    """
    Upload data to a directory bucket.
    :param s3_client: The boto3 S3 client
    :param bucket_name: The bucket that will contain the object
    :param key_name: The key of the object to be uploaded
    :param object_bytes: The data to upload
    """
    try:
        response = s3_client.put_object(Bucket=bucket_name, Key=key_name,
                                Body=object_bytes)
        print(f"Upload object '{key_name}' to bucket '{bucket_name}'.")
        return response
    except ClientError:
        print(f"Couldn't upload object '{key_name}' to bucket '{bucket_name}'.")
        raise

def main():
    # Share the client session with functions and objects to benefit from S3 Express
 One Zone auth key
    s3_client = boto3.client('s3')
    # Directory bucket name must end with --zone-id--x-s3
    resp = put_object(s3_client, 'doc-bucket-example--use1-az5--x-s3', 'sample.txt',
 b'Hello, World!')
    print(resp)

if __name__ == "__main__":
    main()
```

Using the AWS CLI

The following `put-object` example command shows how you can use the AWS CLI to upload an object from Amazon S3. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api put-object --bucket bucket-base-name--zone-id--x-s3 --key sampleinut/
file001.bin --body bucket-seed/file001.bin
```

For more information, see put-object in the *AWS CLI Command Reference*.

Topics

- Using multipart uploads with directory buckets

Using multipart uploads with directory buckets

You can use the multipart upload process to upload a single object as a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently and in any order. If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object. In general, when your object size reaches 100 MB, you should consider using multipart uploads instead of uploading the object in a single operation.

Using multipart upload provides the following advantages:

- **Improved throughput** – You can upload parts in parallel to improve throughput.
- **Quick recovery from any network issues** – Smaller part sizes minimize the impact of restarting a failed upload because of a network error.
- **Pause and resume object uploads** – You can upload object parts over time. After you initiate a multipart upload, there is no expiration date. You must explicitly complete or abort the multipart upload.
- **Begin an upload before you know the final object size** – You can upload an object as you are creating it.

We recommend that you use multipart uploads in the following ways:

- If you're uploading large objects over a stable high-bandwidth network, use multipart uploads to maximize the use of your available bandwidth by uploading object parts in parallel for multi-threaded performance.
- If you're uploading over a spotty network, use multipart uploads to increase resiliency to network errors by avoiding upload restarts. When using multipart uploads, you need to retry uploading only the parts that are interrupted during the upload. You don't need to restart uploading your object from the beginning.

When you're using multipart uploads to upload objects to the Amazon S3 Express One Zone storage class in directory buckets, the multipart upload process is similar to the process of using multipart upload to upload objects to general purpose buckets. However, there are some notable differences.

For more information about using multipart uploads to upload objects to S3 Express One Zone, see the following topics.

Topics

- The multipart upload process
- Checksums with multipart upload operations
- Concurrent multipart upload operations
- Multipart uploads and pricing
- Multipart upload API operations and permissions
- Examples

The multipart upload process

A multipart upload is a three-step process:

- You initiate the upload.
- You upload the object parts.
- After you have uploaded all of the parts, you complete the multipart upload.

Upon receiving the complete multipart upload request, Amazon S3 constructs the object from the uploaded parts, and you can then access the object as you would any other object in your bucket.

Multipart upload initiation

When you send a request to initiate a multipart upload, Amazon S3 returns a response with an upload ID, which is a unique identifier for your multipart upload. You must include this upload ID whenever you upload parts, list the parts, complete an upload, or abort an upload.

Parts upload

When uploading a part, in addition to the upload ID, you must specify a part number. When you're using a multipart upload with S3 Express One Zone, the multipart part numbers must be consecutive part numbers. If you try to complete a multipart upload request with nonconsecutive part numbers, an `HTTP 400 Bad Request` (Invalid Part Order) error is generated.

A part number uniquely identifies a part and its position in the object that you are uploading. If you upload a new part by using the same part number as a previously uploaded part, the previously uploaded part is overwritten.

Whenever you upload a part, Amazon S3 returns an entity tag (ETag) header in its response. For each part upload, you must record the part number and the ETag value. The ETag values for all object part uploads will remain the same, but each part will be assigned a different part number. You must include these values in the subsequent request to complete the multipart upload.

Amazon S3 automatically encrypts all new objects that are uploaded to an S3 bucket. When doing a multipart upload, if you don't specify encryption information in your request, the encryption setting of the uploaded parts is set to the default encryption configuration of the destination bucket. The default encryption configuration of an Amazon S3 bucket is always enabled and is at a minimum set to server-side encryption with Amazon S3 managed keys (SSE-S3). For directory buckets, SSE-S3 and server-side encryption with AWS KMS keys (SSE-KMS) are supported. For more information, see Data protection and encryption.

Multipart upload completion

When you complete a multipart upload, Amazon S3 creates the object by concatenating the parts in ascending order based on the part number. After a successful *complete* request, the parts no longer exist.

Your *complete multipart upload* request must include the upload ID and a list of both part numbers and their corresponding ETag values. The Amazon S3 response includes an ETag that uniquely identifies the combined object data. This ETag is not an MD5 hash of the object data.

Multipart upload listings

You can list the parts of a specific multipart upload or all in-progress multipart uploads. The list parts operation returns the parts information that you have uploaded for a specific multipart upload. For each list parts request, Amazon S3 returns the parts information for the specified multipart upload, up to a maximum of 1,000 parts. If there are more than 1,000 parts in the multipart upload, you must use pagination to retrieve all the parts.

The returned list of parts doesn't include parts that haven't finished uploading. Using the *list multipart uploads* operation, you can obtain a list of multipart uploads that are in progress.

An in-progress multipart upload is an upload that you have initiated, but have not yet completed or aborted. Each request returns at most 1,000 multipart uploads. If there are more than 1,000 multipart uploads in progress, you must send additional requests to retrieve the remaining multipart uploads. Use the returned listing only for verification. Do not use the result of this listing when sending a *complete multipart upload* request. Instead, maintain your own list of the part numbers that you specified when uploading parts and the corresponding ETag values that Amazon S3 returns.

For more information about multipart upload listings, see ListParts in the *Amazon Simple Storage Service API Reference*.

Checksums with multipart upload operations

When you upload an object to, you can specify a checksum algorithm to check object integrity. MD5 is not supported for directory buckets. You can specify one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms:

- CRC32
- CRC32C
- SHA-1
- SHA-256

You can use the Amazon S3 REST API or the AWS SDKs to retrieve the checksum value for individual parts by using `GetObject` or `HeadObject`. If you want to retrieve the checksum values for individual parts of multipart uploads still in process, you can use `ListParts`.

Important

When using the preceding checksum algorithms, the multipart part numbers must use consecutive part numbers. If you try to complete a multipart upload request with

nonconsecutive part numbers, Amazon S3 generates an `HTTP 400 Bad Request` (Invalid Part Order) error.

For more information about how checksums work with multipart upload objects, see Checking object integrity in Amazon S3.

Concurrent multipart upload operations

In a distributed development environment, your application can initiate several updates on the same object at the same time. For example, your application might initiate several multipart uploads by using the same object key. For each of these uploads, your application can then upload parts and send a complete upload request to Amazon S3 to create the object. For S3 Express One Zone, the object creation time is the completion date of the multipart upload.

Important

Versioning isn't supported for objects that are stored in directory buckets.

Multipart uploads and pricing

After you initiate a multipart upload, Amazon S3 retains all the parts until you either complete or abort the upload. Throughout its lifetime, you are billed for all storage, bandwidth, and requests for this multipart upload and its associated parts. If you abort the multipart upload, Amazon S3 deletes the upload artifacts and any parts that you have uploaded, and you are no longer billed for them. There are no early delete charges for deleting incomplete multipart uploads, regardless of the storage class specified. For more information about pricing, see Amazon S3 pricing.

Important

If the complete multipart upload request isn't sent successfully, the object parts aren't assembled and an object isn't created. You are billed for all storage associated with uploaded parts. It's important that you either complete the multipart upload to have the object created or abort the multipart upload to remove any uploaded parts.
Before you can delete a directory bucket, you must complete or abort all in-progress multipart uploads. Directory buckets don't support S3 Lifecycle configurations. If needed,

you can list your active multipart uploads, then abort the uploads, and then delete your bucket.

Multipart upload API operations and permissions

To allow access to object management API operations on a directory bucket, you grant the `s3express:CreateSession` permission in a bucket policy or an AWS Identity and Access Management (IAM) identity-based policy.

You must have the necessary permissions to use the multipart upload operations. You can use bucket policies or IAM identity-based policies to grant IAM principals permissions to perform these operations. The following table lists the required permissions for various multipart upload operations.

You can identify the initiator of a multipart upload through the `Initiator` element. If the initiator is an AWS account, this element provides the same information as the `Owner` element. If the initiator is an IAM user, this element provides the user ARN and display name.

Action	Required permissions
Create a multipart upload	To create the multipart upload, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket.
Initiate a multipart upload	To initiate the multipart upload, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket.
Upload a part	To upload a part, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket. For the initiator to upload a part, the bucket owner must allow the initiator to perform the `s3express:CreateSession` action on the directory bucket.
Upload a part (copy)	To upload a part, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket.

Action	Required permissions
	For the initiator to upload a part for an object, the owner of the bucket must allow the initiator to perform the `s3express:CreateSession` action on the object.
Complete a multipart upload	To complete a multipart upload, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket. For the initiator to complete a multipart upload, the bucket owner must allow the initiator to perform the `s3express:CreateSession` action on the object.
Abort a multipart upload	To abort a multipart upload, you must be allowed to perform the `s3express:CreateSession` action. For the initiator to abort a multipart upload, the initiator must be granted explicit allow access to perform the `s3express:CreateSession` action.
List parts	To list the parts in a multipart upload, you must be allowed to perform the `s3express:CreateSession` action on the directory bucket.
List in-progress multipart uploads	To list the in-progress multipart uploads to a bucket, you must be allowed to perform the `s3:ListBucketMultipartUploads` action on that bucket.

API operation support for multipart uploads

The following sections in the Amazon Simple Storage Service API Reference describe the Amazon S3 REST API operations for multipart uploads.

- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- AbortMultipartUpload
- ListParts

- ListMultipartUploads

Examples

To use a multipart upload to upload an object to S3 Express One Zone in a directory bucket, see the following examples.

Topics

- Creating a multipart upload
- Uploading the parts of a multipart upload
- Completing a multipart upload
- Aborting a multipart upload
- Creating a multipart upload copy operation
- Listing in-progress multipart uploads
- Listing the parts of a multipart upload

Creating a multipart upload

> **Note**
>
> For directory buckets, when you perform a `CreateMultipartUpload` operation and an `UploadPartCopy` operation, the bucket's default encryption must use the desired encryption configuration, and the request headers you provide in the `CreateMultipartUpload` request must match the default encryption configuration of the destination bucket.

The following examples show how to create a multipart upload.

Using the AWS SDKs

SDK for Java 2.x

Example

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
```

```
 * all the upload parts
 *
 * @param s3
 * @param bucketName - for example, 'doc-example-bucket--use1-az4--x-s3'
 * @param key
 * @return
 */
 private static String createMultipartUpload(S3Client s3, String bucketName, String
 key) {

     CreateMultipartUploadRequest createMultipartUploadRequest =
 CreateMultipartUploadRequest.builder()
             .bucket(bucketName)
             .key(key)
             .build();

     String uploadId = null;

     try {
         CreateMultipartUploadResponse response =
 s3.createMultipartUpload(createMultipartUploadRequest);
         uploadId = response.uploadId();
     }
     catch (S3Exception e) {
         System.err.println(e.awsErrorDetails().errorMessage());
         System.exit(1);
     }
     return uploadId;
```

SDK for Python (Boto3)

Example

```
def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
   Create a multipart upload to a directory bucket

   :param s3_client: boto3 S3 client
   :param bucket_name: The destination bucket for the multipart upload
   :param key_name: The key name for the object to be uploaded
   :return: The UploadId for the multipart upload if created successfully, else None
   '''

   try:
```

```
        mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
 key_name)
        return mpu['UploadId']
    except ClientError as e:
        logging.error(e)
        return None
```

Using the AWS CLI

This example shows how to create a multipart upload to a directory bucket by using the AWS CLI. This command starts a multipart upload to the directory bucket *bucket-base-name*--*zone-id*--x-s3 for the object *KEY_NAME*. To use the command replace the *user input placeholders* with your own information.

```
aws s3api create-multipart-upload --bucket bucket-base-name--zone-id--x-s3 --
key KEY_NAME
```

For more information, see create-multipart-upload in the AWS Command Line Interface.

Uploading the parts of a multipart upload

The following examples show how to upload parts of a multipart upload.

Using the AWS SDKs

SDK for Java 2.x

The following example shows how to break a single object into parts and then upload those parts to a directory bucket by using the SDK for Java 2.x.

Example

```
/**
 * This method creates part requests and uploads individual parts to S3 and then
 returns all the completed parts
 *
 * @param s3
 * @param bucketName
 * @param key
 * @param uploadId
 * @throws IOException
 */
```

```
private static ListCompletedPartmultipartUpload(S3Client s3, String bucketName,
String key, String uploadId, String filePath) throws IOException {

        int partNumber = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // read the local file, breakdown into chunks and process
        try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
            long fileSize = file.length();
            int position = 0;
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer.
                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .partNumber(partNumber)
                        .build();

                UploadPartResponse partResponse = s3.uploadPart(
                        uploadPartRequest,
                        RequestBody.fromByteBuffer(bb));

                CompletedPart part = CompletedPart.builder()
                        .partNumber(partNumber)
                        .eTag(partResponse.eTag())
                        .build();
                completedParts.add(part);

                bb.clear();
                position += read;
                partNumber++;
            }
        }

        catch (IOException e) {
            throw e;
        }
        return completedParts;
```

```
    }
```

SDK for Python (Boto3)

The following example shows how to break a single object into parts and then upload those parts to a directory bucket by using the AWS SDK for Python (Boto3).

Example

```
def multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_size):
    '''
    Break up a file into multiple parts and upload those parts to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name for object to be uploaded and for the local file
 that's being uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
 last part of your multipart upload.
    :return: part_list for the multipart upload if all parts are uploaded
 successfully, else None
    '''

    part_list = []
    try:
        with open(key_name, 'rb') as file:
            part_counter = 1
            while True:
                file_part = file.read(part_size)
                if not len(file_part):
                    break
                upload_part = s3_client.upload_part(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = mpu_id,
                    Body = file_part,
                    PartNumber = part_counter
                )
                part_list.append({'PartNumber': part_counter, 'ETag':
 upload_part['ETag']})
                part_counter += 1
    except ClientError as e:
```

```
            logging.error(e)
            return None
    return part_list
```

Using the AWS CLI

This example shows how to break a single object into parts and then upload those parts to a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api upload-part --bucket bucket-base-name--zone-id--x-s3 --
key KEY_NAME --part-number 1 --body LOCAL_FILE_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

For more information, see upload-part in the AWS Command Line Interface.

Completing a multipart upload

The following examples show how to complete a multipart upload.

Using the AWS SDKs

SDK for Java 2.x

The following examples show how to complete a multipart upload by using the SDK for Java 2.x.

Example

```
/**
 * This method completes the multipart upload request by collating all the upload
 parts
 * @param s3
 * @param bucketName - for example, 'doc-example-bucket--usw2-az1--x-s3'
 * @param key
 * @param uploadId
 * @param uploadParts
 */
 private static void completeMultipartUpload(S3Client s3, String bucketName, String
 key, String uploadId, ListCompletedPart uploadParts) {
        CompletedMultipartUpload completedMultipartUpload =
 CompletedMultipartUpload.builder()
```

```
                .parts(uploadParts)
                .build();

        CompleteMultipartUploadRequest completeMultipartUploadRequest =
                CompleteMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(key)
                        .uploadId(uploadId)
                        .multipartUpload(completedMultipartUpload)
                        .build();

        s3.completeMultipartUpload(completeMultipartUploadRequest);
    }

    public static void multipartUploadTest(S3Client s3, String bucketName, String
key, String localFilePath)  {
        System.out.println("Starting multipart upload for: " + key);
        try {
            String uploadId = createMultipartUpload(s3, bucketName, key);
            System.out.println(uploadId);
            ListCompletedPart parts = multipartUpload(s3, bucketName, key, uploadId,
localFilePath);
            completeMultipartUpload(s3, bucketName, key, uploadId, parts);
            System.out.println("Multipart upload completed for: " + key);
        }

        catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python (Boto3)

The following examples show how to complete a multipart upload by using the SDK for Python (Boto3).

Example

```
def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
```

```
    :param bucket_name: The destination bucket for the multipart upload
    :param key_name: The key name for the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_list: The list of uploaded part numbers with their associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    MB = 1024 ** 2
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'OBJECT_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, bucket_name, key_name)
    if mpu_id is not None:
        part_list = multipart_upload(s3_client, bucket_name, key_name, mpu_id,
 part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id,
 part_list):
                print (f'{key_name} successfully uploaded through a ultipart upload
 to {bucket_name}')
            else:
                print (f'Could not upload {key_name} hrough a multipart upload to
 {bucket_name}')
```

Using the AWS CLI

This example shows how to complete a multipart upload for a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api complete-multipart-upload --bucket bucket-base-name--zone-id--x-s3 --
key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
 --multipart-upload file://parts.json
```

This example takes a JSON structure that describes the parts of the multipart upload that should be reassembled into the complete file. In this example, the `file://` prefix is used to load the JSON structure from a file in the local folder named `parts`.

parts.json:

```
parts.json
{
  "Parts": [
    {
      "ETag": "6b78c4a64dd641a58dac8d9258b88147",
      "PartNumber": 1
    }
  ]
}
```

For more information, see complete-multipart-upload in the AWS Command Line Interface.

Aborting a multipart upload

The following examples show how to abort a multipart upload.

Using the AWS SDKs

SDK for Java 2.x

The following example shows how to abort a multipart upload by using the SDK for Java 2.x.

Example

```
public static void abortMultiPartUploads( S3Client s3, String bucketName ) {

        try {
```

```
            ListMultipartUploadsRequest listMultipartUploadsRequest =
 ListMultipartUploadsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListMultipartUploadsResponse response =
 s3.listMultipartUploads(listMultipartUploadsRequest);
            List<MultipartUpload> uploads = response.uploads();

            AbortMultipartUploadRequest abortMultipartUploadRequest;
            for (MultipartUpload upload: uploads) {
                abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                        .bucket(bucketName)
                        .key(upload.key())
                        .uploadId(upload.uploadId())
                        .build();

                s3.abortMultipartUpload(abortMultipartUploadRequest);
            }

        }

        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python (Boto3)

The following example shows how to abort a multipart upload by using the SDK for Python (Boto3).

Example

```
import logging
import boto3
from botocore.exceptions import ClientError


def abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
    '''
    Aborts a partial multipart upload in a directory bucket.
```

```
    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket where the multipart upload was initiated - for
 example, 'doc-example-bucket--usw2-az1--x-s3'
    :param key_name: Name of the object for which the multipart upload needs to be
 aborted
    :param upload_id: Multipart upload ID for the multipart upload to be aborted
    :return: True if the multipart upload was successfully aborted, False if not
    '''
    try:
        s3_client.abort_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = upload_id
        )
    except ClientError as e:
        logging.error(e)
        return False
    return True


if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
        upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
    if abort_multipart_upload(s3_client, bucket_name, key_name, upload_id):
        print (f'Multipart upload for object {key_name} in {bucket_name} bucket has
 been aborted')
    else:
        print (f'Unable to abort multipart upload for object {key_name} in
 {bucket_name} bucket')
```

Using the AWS CLI

The following example shows how to abort a multipart upload by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api abort-multipart-upload --bucket bucket-base-name--zone-id--x-s3 --
key KEY_NAME --upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
MAQAAAAB0OxUFeA7LTbWWFS8WYwhrxDxTIDN-pdEEq_agIHqsbg"
```

For more information, see abort-multipart-upload in the AWS Command Line Interface.

Creating a multipart upload copy operation

> **Note**
>
> - To encrypt new object part copies in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default encryption configuration with a KMS key (specifically, a customer managed key). The AWS managed key (`aws/s3`) isn't supported. Your SSE-KMS configuration can only support 1 customer managed key per directory bucket for the lifetime of the bucket. After you specify a customer managed key for SSE-KMS, you can't override the customer managed key for the bucket's SSE-KMS configuration. You can't specify server-side encryption settings for new object part copies with SSE-KMS in the UploadPartCopy request headers. Also, the request headers you provide in the `CreateMultipartUpload` request must match the default encryption configuration of the destination bucket.
> - S3 Bucket Keys aren't supported, when you copy SSE-KMS encrypted objects from general purpose buckets to directory buckets, from directory buckets to general purpose buckets, or between directory buckets, through UploadPartCopy. In this case, Amazon S3 makes a call to AWS KMS every time a copy request is made for a KMS-encrypted object.

The following examples show how to copy objects from one bucket to another using a multipart upload.

Using the AWS SDKs

SDK for Java 2.x

The following example shows how to use a multipart upload to programmatically copy an object from one bucket to another by using the SDK for Java 2.x.

Example

```
/**
 * This method creates a multipart upload request that generates a unique upload ID
 that is used to track
 * all the upload parts.
 *
 * @param s3
```

```
* @param bucketName
* @param key
* @return
*/
private static String createMultipartUpload(S3Client s3, String bucketName, String
key) {
       CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
               .bucket(bucketName)
               .key(key)
               .build();
       String uploadId = null;
       try {
           CreateMultipartUploadResponse response =
s3.createMultipartUpload(createMultipartUploadRequest);
           uploadId = response.uploadId();
       } catch (S3Exception e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
       return uploadId;
 }

 /**
  * Creates copy parts based on source object size and copies over individual parts
  *
  * @param s3
  * @param sourceBucket
  * @param sourceKey
  * @param destnBucket
  * @param destnKey
  * @param uploadId
  * @return
  * @throws IOException
  */
   public static ListCompletedPart multipartUploadCopy(S3Client s3, String
sourceBucket, String sourceKey, String destnBucket, String destnKey, String
uploadId) throws IOException {

       // Get the object size to track the end of the copy operation.
       HeadObjectRequest headObjectRequest = HeadObjectRequest
               .builder()
               .bucket(sourceBucket)
               .key(sourceKey)
```

```
                .build();
        HeadObjectResponse response = s3.headObject(headObjectRequest);
        Long objectSize = response.contentLength();

        System.out.println("Source Object size: " + objectSize);

        // Copy the object using 20 MB parts.
        long partSize = 20 * 1024 * 1024;
        long bytePosition = 0;
        int partNum = 1;
        ListCompletedPart completedParts = new ArrayList<>();
        while (bytePosition < objectSize) {
            // The last part might be smaller than partSize, so check to make sure
            // that lastByte isn't beyond the end of the object.
            long lastByte = Math.min(bytePosition + partSize - 1, objectSize - 1);

            System.out.println("part no: " + partNum + ", bytePosition: " +
bytePosition + ", lastByte: " + lastByte);

            // Copy this part.
            UploadPartCopyRequest req = UploadPartCopyRequest.builder()
                    .uploadId(uploadId)
                    .sourceBucket(sourceBucket)
                    .sourceKey(sourceKey)
                    .destinationBucket(destnBucket)
                    .destinationKey(destnKey)
                    .copySourceRange("bytes="+bytePosition+"-"+lastByte)
                    .partNumber(partNum)
                    .build();
            UploadPartCopyResponse res = s3.uploadPartCopy(req);
            CompletedPart part = CompletedPart.builder()
                    .partNumber(partNum)
                    .eTag(res.copyPartResult().eTag())
                    .build();
            completedParts.add(part);
            partNum++;
            bytePosition += partSize;
        }
        return completedParts;
    }


    public static void multipartCopyUploadTest(S3Client s3, String srcBucket, String
srcKey, String destnBucket, String destnKey)  {
```

```
        System.out.println("Starting multipart copy for: " + srcKey);
        try {
            String uploadId = createMultipartUpload(s3, destnBucket, destnKey);
            System.out.println(uploadId);
            ListCompletedPart parts = multipartUploadCopy(s3, srcBucket,
 srcKey,destnBucket,  destnKey, uploadId);
            completeMultipartUpload(s3, destnBucket, destnKey, uploadId, parts);
            System.out.println("Multipart copy completed for: " + srcKey);
        } catch (Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

SDK for Python (Boto3)

The following example shows how to use a multipart upload to programmatically copy an object from one bucket to another by using the SDK for Python (Boto3).

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def head_object(s3_client, bucket_name, key_name):
    '''
    Returns metadata for an object in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains the object to query for metadata
    :param key_name: Key name to query for metadata
    :return: Metadata for the specified object if successful, else None
    '''

    try:
        response = s3_client.head_object(
            Bucket = bucket_name,
            Key = key_name
        )
        return response
    except ClientError as e:
        logging.error(e)
        return None
```

```
def create_multipart_upload(s3_client, bucket_name, key_name):
    '''
    Create a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :return: UploadId for the multipart upload if created successfully, else None
    '''

    try:
        mpu = s3_client.create_multipart_upload(Bucket = bucket_name, Key =
 key_name)
        return mpu['UploadId']
    except ClientError as e:
        logging.error(e)
        return None

def multipart_copy_upload(s3_client, source_bucket_name, key_name,
 target_bucket_name, mpu_id, part_size):
    '''
    Copy an object in a directory bucket to another bucket in multiple parts of a
 specified size

    :param s3_client: boto3 S3 client
    :param source_bucket_name: Bucket where the source object exists
    :param key_name: Key name of the object to be copied
    :param target_bucket_name: Destination bucket for copied object
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_size: The size parts that the object will be broken into, in bytes.
                      Minimum 5 MiB, Maximum 5 GiB. There is no minimum size for the
 last part of your multipart upload.
    :return: part_list for the multipart copy if all parts are copied successfully,
 else None
    '''

    part_list = []
    copy_source = {
        'Bucket': source_bucket_name,
        'Key': key_name
    }
    try:
        part_counter = 1
```

```
        object_size = head_object(s3_client, source_bucket_name, key_name)
        if object_size is not None:
            object_size = object_size['ContentLength']
        while (part_counter - 1) * part_size <object_size:
            bytes_start = (part_counter - 1) * part_size
            bytes_end = (part_counter * part_size) - 1
            upload_copy_part = s3_client.upload_part_copy (
                Bucket = target_bucket_name,
                CopySource = copy_source,
                CopySourceRange = f'bytes={bytes_start}-{bytes_end}',
                Key = key_name,
                PartNumber = part_counter,
                UploadId = mpu_id
            )
            part_list.append({'PartNumber': part_counter, 'ETag':
 upload_copy_part['CopyPartResult']['ETag']})
            part_counter += 1
    except ClientError as e:
        logging.error(e)
        return None
    return part_list

def complete_multipart_upload(s3_client, bucket_name, key_name, mpu_id, part_list):
    '''
    Completes a multipart upload to a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Destination bucket for the multipart upload
    :param key_name: Key name of the object to be uploaded
    :param mpu_id: The UploadId returned from the create_multipart_upload call
    :param part_list: List of uploaded part numbers with associated ETags
    :return: True if the multipart upload was completed successfully, else False
    '''

    try:
        s3_client.complete_multipart_upload(
            Bucket = bucket_name,
            Key = key_name,
            UploadId = mpu_id,
            MultipartUpload = {
                'Parts': part_list
            }
        )
    except ClientError as e:
```

```
        logging.error(e)
        return False
    return True

if __name__ == '__main__':
    MB = 1024 ** 2
    region = 'us-west-2'
    source_bucket_name = 'SOURCE_BUCKET_NAME'
    target_bucket_name = 'TARGET_BUCKET_NAME'
    key_name = 'KEY_NAME'
    part_size = 10 * MB
    s3_client = boto3.client('s3', region_name = region)
    mpu_id = create_multipart_upload(s3_client, target_bucket_name, key_name)
    if mpu_id is not None:
        part_list = multipart_copy_upload(s3_client, source_bucket_name, key_name,
 target_bucket_name, mpu_id, part_size)
        if part_list is not None:
            if complete_multipart_upload(s3_client, target_bucket_name, key_name,
 mpu_id, part_list):
                print (f'{key_name} successfully copied through multipart copy from
 {source_bucket_name} to {target_bucket_name}')
            else:
                print (f'Could not copy {key_name} through multipart copy from
 {source_bucket_name} to {target_bucket_name}')
```

Using the AWS CLI

The following example shows how to use a multipart upload to programmatically copy an object from one bucket to a directory bucket using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api upload-part-copy --bucket bucket-base-name--zone-id--x-s3 --
key TARGET_KEY_NAME --copy-source SOURCE_BUCKET_NAME/SOURCE_KEY_NAME --part-number 1 --
upload-id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

For more information, see upload-part-copy in the AWS Command Line Interface.

Listing in-progress multipart uploads

To list in-progress multipart uploads to a directory bucket, you can use the AWS SDKs, or the AWS CLI.

Using the AWS SDKs

SDK for Java 2.x

The following examples show how to list in-progress (incomplete) multipart uploads by using the SDK for Java 2.x.

Example

```
public static void listMultiPartUploads( S3Client s3, String bucketName) {
       try {
           ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

           ListMultipartUploadsResponse response =
s3.listMultipartUploads(listMultipartUploadsRequest);
           List MultipartUpload uploads = response.uploads();
           for (MultipartUpload upload: uploads) {
               System.out.println("Upload in progress: Key = \"" + upload.key() +
"\", id = " + upload.uploadId());
           }
     }
     catch (S3Exception e) {
           System.err.println(e.getMessage());
           System.exit(1);
     }
 }
```

SDK for Python (Boto3)

The following examples show how to list in-progress (incomplete) multipart uploads by using the SDK for Python (Boto3).

Example

```
import logging
import boto3
from botocore.exceptions import ClientError

def list_multipart_uploads(s3_client, bucket_name):
```

```
    '''
    List any incomplete multipart uploads in a directory bucket in e specified gion

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket to check for incomplete multipart uploads
    :return: List of incomplete multipart uploads if there are any, None if not
    '''

    try:
        response = s3_client.list_multipart_uploads(Bucket = bucket_name)
        if 'Uploads' in response.keys():
            return response['Uploads']
        else:
            return None
    except ClientError as e:
        logging.error(e)

if __name__ == '__main__':
    bucket_name = 'BUCKET_NAME'
    region = 'us-west-2'
    s3_client = boto3.client('s3', region_name = region)
    multipart_uploads = list_multipart_uploads(s3_client, bucket_name)
    if multipart_uploads is not None:
        print (f'There are {len(multipart_uploads)} ncomplete multipart uploads for
 {bucket_name}')
    else:
        print (f'There are no incomplete multipart uploads for {bucket_name}')
```

Using the AWS CLI

The following examples show how to list in-progress (incomplete) multipart uploads by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

```
aws s3api list-multipart-uploads --bucket bucket-base-name--zone-id--x-s3
```

For more information, see list-multipart-uploads in the AWS Command Line Interface.

Listing the parts of a multipart upload

The following examples show how to list the parts of a multipart upload to a directory bucket.

Using the AWS SDKs

SDK for Java 2.x

The following examples show how to list the parts of a multipart upload to a directory bucket by using SDK for Java 2.x.

```
public static void listMultiPartUploadsParts( S3Client s3, String bucketName, String
 objKey, String uploadID) {

        try {
            ListPartsRequest listPartsRequest = ListPartsRequest.builder()
                .bucket(bucketName)
                .uploadId(uploadID)
                .key(objKey)
                .build();

            ListPartsResponse response = s3.listParts(listPartsRequest);
            ListPart parts = response.parts();
            for (Part part: parts) {
                System.out.println("Upload in progress: Part number = \"" +
 part.partNumber() + "\", etag = " + part.eTag());
            }

        }

        catch (S3Exception e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }


    }
```

SDK for Python (Boto3)

The following examples show how to list the parts of a multipart upload to a directory bucket by using SDK for Python (Boto3).

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
def list_parts(s3_client, bucket_name, key_name, upload_id):
    '''
    Lists the parts that have been uploaded for a specific multipart upload to a
 directory bucket.

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that multipart uploads parts have been uploaded to
    :param key_name: Name of the object that has parts uploaded
    :param upload_id: Multipart upload ID that the parts are associated with
    :return: List of parts associated with the specified multipart upload, None if
 there are no parts
    '''
    parts_list = []
    next_part_marker = ''
    continuation_flag = True
    try:
        while continuation_flag:
            if next_part_marker == '':
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id
                )
            else:
                response = s3_client.list_parts(
                    Bucket = bucket_name,
                    Key = key_name,
                    UploadId = upload_id,
                    NextPartMarker = next_part_marker
                )
            if 'Parts' in response:
                for part in response['Parts']:
                    parts_list.append(part)
                if response['IsTruncated']:
                    next_part_marker = response['NextPartNumberMarker']
                else:
                    continuation_flag = False
            else:
                continuation_flag = False
        return parts_list
    except ClientError as e:
        logging.error(e)
        return None
```

```
if __name__ == '__main__':
    region = 'us-west-2'
    bucket_name = 'BUCKET_NAME'
    key_name = 'KEY_NAME'
    upload_id = 'UPLOAD_ID'
    s3_client = boto3.client('s3', region_name = region)
    parts_list = list_parts(s3_client, bucket_name, key_name, upload_id)
    if parts_list is not None:
        print (f'{key_name} has {len(parts_list)} parts uploaded to {bucket_name}')
    else:
        print (f'There are no multipart uploads with that upload ID for
 {bucket_name} bucket')
```

Using the AWS CLI

The following examples show how to list the parts of a multipart upload to a directory bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3api list-parts --bucket bucket-base-name--zone-id--x-s3 --key KEY_NAME --upload-
id
 "AS_mgt9RaQE9GEaifATue15dAAAAAAAAAAAEMAAAAAAAAADQwNzI4MDU0MjUyMBYAAAAAAAAAAA0AAAAAAAAAAH2AfYAA
```

For more information, see list-parts in the AWS Command Line Interface.

Copying objects from or to a directory bucket

The copy operation creates a copy of an object that is already stored in Amazon S3. You can copy objects between directory buckets and general purpose buckets. You can also copy objects within a bucket and across buckets of the same type, for example, from directory bucket to directory bucket.

> **Note**
>
> Copying objects across different AWS Regions isn't supported when the source or destination bucket is in an AWS Local Zone. The source and destination buckets must have the same parent AWS Region. The source and destination buckets can be different bucket location types (Availability Zone or Local Zone).

You can create a copy of object up to 5 GB in a single atomic operation. However, to copy an object that is greater than 5 GB, you must use the multipart upload API operations. For more information, see Using multipart uploads with directory buckets.

Permissions

To copy objects, you must have the following permissions:

- To copy objects from one directory bucket to another directory bucket, you must have the `s3express:CreateSession` permission.
- To copy objects from directory buckets to general purpose buckets, you must have the `s3express:CreateSession` permission and the `s3:PutObject` permission to write the object copy to the destination bucket.
- To copy objects from general purpose buckets to directory buckets, you must have the `s3express:CreateSession` permission and `s3:GetObject` permission to read the source object that is being copied.

  For more information, see CopyObject in the *Amazon Simple Storage Service API Reference*.

Encryption

Amazon S3 automatically encrypts all new objects that are uploaded to an S3 bucket. The default encryption configuration of an S3 bucket is always enabled and is at a minimum set to server-side encryption with Amazon S3 managed keys (SSE-S3).

For directory buckets, SSE-S3 and server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) are supported. When the destination bucket is a directory bucket, we recommend that the destination bucket's default encryption uses the desired encryption configuration and you don't override the bucket default encryption. Then, new objects are automatically encrypted with the desired encryption settings. Also, S3 Bucket Keys aren't supported, when you copy SSE-KMS encrypted objects from general purpose buckets to directory buckets, from directory buckets to general purpose buckets, or between directory buckets, through CopyObject. In this case, Amazon S3 makes a call to AWS KMS every time a copy request is made for a KMS-encrypted object. For more information about the encryption overriding behaviors in directory buckets, see Specifying server-side encryption with AWS KMS for new object uploads.

For general purpose buckets, you can use SSE-S3 (the default), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C).

If you make a copy request that specifies to use DSSE-KMS or SSE-C for a directory bucket (either the source or destination bucket), the response returns an error.

Tags

Directory buckets don't support tags. If you copy an object that has tags from a general purpose bucket to a directory bucket, you receive an HTTP `501 (Not Implemented)` response. For more information, see CopyObject in the *Amazon Simple Storage Service API Reference*.

ETags

Entity tags (ETags) for S3 Express One Zone are random alphanumeric strings and are not MD5 checksums. To help ensure object integrity, use additional checksums.

Additional checksums

S3 Express One Zone offers you the option to choose the checksum algorithm that is used to validate your data during upload or download. You can select one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms: CRC32, CRC32C, SHA-1, and SHA-256. MD5-based checksums are not supported with the S3 Express One Zone storage class.

For more information, see S3 additional checksum best practices.

Supported features

For more information about which Amazon S3 features are supported for S3 Express One Zone, see Differences for directory buckets.

Using the S3 console (copy to a directory bucket)

> **Note**
>
> The restrictions and limitations when you copy an object to a directory bucket with the console are as follows:
>
> - The Copy action applies to all objects within the specified folders (prefixes). Objects added to these folders while the action is in progress might be affected.
> - Objects encrypted with customer-provided encryption keys (SSE-C) cannot be copied by using the S3 console. To copy objects encrypted with SSE-C, use the AWS CLI, AWS SDK, or the Amazon S3 REST API.

- Copied objects will not retain the Object Lock settings from the original objects.
- If the bucket you are copying objects from uses the bucket owner enforced setting for S3 Object Ownership, object ACLs will not be copied to the specified destination.
- If you want to copy objects to a bucket that uses the bucket owner enforced setting for S3 Object Ownership, make sure that the source bucket also uses the bucket owner enforced setting, or remove any object ACL grants to other AWS accounts and groups.
- Objects copied from a general purpose bucket to a directory bucket will not retain object tags, ACLs, or Etag values. Checksum values can be copied, but are not equivalent to an Etag. The checksum value may change compared to when it was added.
- All objects copied to a directory bucket will be with the bucket owner enforced setting for S3 Object Ownership.

To copy an object from a general purpose bucket or a directory bucket to a directory bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, the bucket type that you want to copy objects from:
   - To copy from a general purpose bucket, choose the **General purpose buckets** tab.
   - To copy from a directory bucket, choose the **Directory buckets** tab.
3. Choose the general purpose bucket or directory bucket that contains the objects that you want to copy.
4. Choose the **Objects** tab. On the **Objects** page, select the check box to the left of the names of the objects that you want to copy.
5. On the **Actions** menu, choose **Copy**.

   The **Copy** page appears.
6. Under **Destination**, choose **Directory bucket** for your destination type. To specify the destination path, choose **Browse S3**, navigate to the destination, and then choose the option button to the left of the destination. Choose **Choose destination**.

   Alternatively, enter the destination path.
7. Under **Additional copy settings**, choose whether you want to **Copy source settings**, **Don't specify settings**, or **Specify settings**. **Copy source settings** is the default option. If you

only want to copy the object without the source settings attributes, choose **Don't specify settings**. Choose **Specify settings** to specify settings for server-side encryption, checksums, and metadata.

8. Choose **Copy**. Amazon S3 copies your objects to the destination.

Using the S3 console (copy to a general purpose bucket)

> **Note**
>
> The restrictions and limitations when you copy an object to a general purpose bucket with the console are as follows:
>
> - The Copy action applies to all objects within the specified folders (prefixes). Objects added to these folders while the action is in progress might be affected.
> - Objects encrypted with customer-provided encryption keys (SSE-C) cannot be copied by using the S3 console. To copy objects encrypted with SSE-C, use the AWS CLI, AWS SDK, or the Amazon S3 REST API.
> - Copied objects will not retain the Object Lock settings from the original objects.
> - If the bucket you are copying objects from uses the bucket owner enforced setting for S3 Object Ownership, object ACLs will not be copied to the specified destination.
> - If you want to copy objects to a bucket that uses the bucket owner enforced setting for S3 Object Ownership, make sure that the source bucket also uses the bucket owner enforced setting, or remove any object ACL grants to other AWS accounts and groups.

To copy an object from a directory bucket to a general purpose bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the **Directory buckets** tab.
4. Choose the directory bucket that contains the objects that you want to copy.
5. Choose the **Objects** tab. On the **Objects** page, select the check box to the left of the names of the objects that you want to copy.
6. On the **Actions** menu, choose **Copy**.

7. Under **Destination**, choose **General purpose bucket** for your destination type. To specify the destination path, choose **Browse S3**, navigate to the destination, and choose the option button to the left of the destination. Choose **Choose destination**.

   Alternatively, enter the destination path.

8. Under **Additional copy settings**, choose whether you want to **Copy source settings**, **Don't specify settings**, or **Specify settings**. **Copy source settings** is the default option. If you only want to copy the object without the source settings attributes, choose **Don't specify settings**. Choose **Specify settings** to specify settings for storage class, ACLs, object tags, metadata, server-side encryption, and additional checksums.
9. Choose **Copy**. Amazon S3 copies your objects to the destination.

Using the AWS SDKs

SDK for Java 2.x

Example

```
public static void copyBucketObject (S3Client s3, String sourceBucket, String
objectKey, String targetBucket) {
     CopyObjectRequest copyReq = CopyObjectRequest.builder()
         .sourceBucket(sourceBucket)
         .sourceKey(objectKey)
         .destinationBucket(targetBucket)
         .destinationKey(objectKey)
         .build();
      String temp = "";

      try {
          CopyObjectResponse copyRes = s3.copyObject(copyReq);
          System.out.println("Successfully copied " + objectKey +" from bucket " +
sourceBucket +" into bucket "+targetBucket);
      }

      catch (S3Exception e) {
          System.err.println(e.awsErrorDetails().errorMessage());
          System.exit(1);
      }
}
```

Using the AWS CLI

The following `copy-object` example command shows how you can use the AWS CLI to copy an object from one bucket to another bucket. You can copy objects between bucket types. To run this command, replace the user input placeholders with your own information.

```
aws s3api copy-object --copy-source SOURCE_BUCKET/SOURCE_KEY_NAME --key TARGET_KEY_NAME
 --bucket TARGET_BUCKET_NAME
```

For more information, see copy-object in the *AWS CLI Command Reference*.

Deleting objects from a directory bucket

You can delete objects from an Amazon S3 directory bucket by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), or AWS SDKs. For more information, see Working with directory buckets and S3 Express One Zone.

Warning

- Deleting an object can't be undone.
- This action deletes all specified objects. When deleting folders, wait for the delete action to finish before adding new objects to the folder. Otherwise, new objects might be deleted as well.

Note

When you programmatically delete multiple objects from a directory bucket, note the following:

- Object keys in `DeleteObjects` requests must contain at least one non-white space character. Strings of all white space characters are not supported.
- Object keys in `DeleteObjects` requests cannot contain Unicode control characters, except for newline (`\n`), tab (`\t`), and carriage return (`\r`).

Using the S3 console

To delete objects

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Directory buckets**.
3. Choose the directory bucket that contains the objects that you want to delete.
4. Choose the **Objects** tab. In the **Objects** list, select the check box to the left of the object or objects that you want to delete.
5. Choose **Delete**.
6. On the **Delete objects** page, enter **`permanently delete`** in the text box.
7. Choose **Delete objects**.

Using the AWS SDKs

SDK for Java 2.x

Example

The following example deletes objects in a directory bucket by using the AWS SDK for Java 2.x.

```
static void deleteObject(S3Client s3Client, String bucketName, String objectKey) {



        try {

            DeleteObjectRequest del = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            s3Client.deleteObject(del);

            System.out.println("Object " + objectKey + " has been deleted");


        } catch (S3Exception e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

    }
```

SDK for Python (Boto3)

Example

The following example deletes objects in a directory bucket by using the AWS SDK for Python (Boto3).

```
import logging
import boto3
from botocore.exceptions import ClientError

def delete_objects(s3_client, bucket_name, objects):
    '''
    Delete a list of objects in a directory bucket

    :param s3_client: boto3 S3 client
    :param bucket_name: Bucket that contains objects to be deleted; for example,
 'doc-example-bucket--usw2-az1--x-s3'
    :param objects: List of dictionaries that specify the key names to delete
    :return: Response output, else False
    '''

    try:
        response = s3_client.delete_objects(
            Bucket = bucket_name,
            Delete = {
                'Objects': objects
            }
        )
        return response
    except ClientError as e:
        logging.error(e)
        return False


if __name__ == '__main__':
    region = 'us-west-2'
```

```
    bucket_name = 'BUCKET_NAME'
    objects = [
        {
            'Key': '0.txt'
        },
        {
            'Key': '1.txt'
        },
        {
            'Key': '2.txt'
        },
        {
            'Key': '3.txt'
        },
        {
            'Key': '4.txt'
        }
    ]

    s3_client = boto3.client('s3', region_name = region)
    results = delete_objects(s3_client, bucket_name, objects)
    if results is not None:
        if 'Deleted' in results:
            print (f'Deleted {len(results["Deleted"])} objects from {bucket_name}')
        if 'Errors' in results:
            print (f'Failed to delete {len(results["Errors"])} objects from
{bucket_name}')
```

Using the AWS CLI

The following `delete-object` example command shows how you can use the AWS CLI to delete an object from a directory bucket. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api delete-object --bucket bucket-base-name--zone-id--x-s3 --key KEY_NAME
```

For more information, see delete-object in the *AWS CLI Command Reference*.

The following `delete-objects` example command shows how you can use the AWS CLI to delete objects from a directory bucket. To run this command, replace the *`user input placeholders`* with your own information.

The `delete.json` file is as follows:

```
{
    "Objects": [
        {
            "Key": "0.txt"
        },
        {
            "Key": "1.txt"
        },
        {
            "Key": "2.txt"
        },
        {
            "Key": "3.txt"
        }
    ]
}
```

The `delete-objects` example command is as follows:

```
aws s3api delete-objects --bucket bucket-base-name--zone-id--x-s3 --delete
 file://delete.json
```

For more information, see delete-objects in the *AWS CLI Command Reference*.

Downloading an object from a directory bucket

The following code examples show how to read data from (download) an object in an Amazon S3 directory bucket by using the `GetObject` API operation.

Using the AWS SDKs

SDK for Java 2.x

Example

The following code example shows how to read data from an object in a directory bucket by using the AWS SDK for Java 2.x.

```
public static void getObject(S3Client s3Client, String bucketName, String objectKey)
 {
     try {
         GetObjectRequest objectRequest = GetObjectRequest
            .builder()
            .key(objectKey)
            .bucket(bucketName)
            .build();

         ResponseBytes GetObjectResponse objectBytes =
 s3Client.getObjectAsBytes(objectRequest);
         byte[] data = objectBytes.asByteArray();

         //Print object contents to console
         String s = new String(data, StandardCharsets.UTF_8);
         System.out.println(s);
    }

    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

SDK for Python (Boto3)

Example

The following code example shows how to read data from an object in a directory bucket by using the AWS SDK for Python (Boto3).

```
import boto3
from botocore.exceptions import ClientError
from botocore.response import StreamingBody

def get_object(s3_client: boto3.client, bucket_name: str, key_name: str) ->
 StreamingBody:
    """
    Gets the object.
    :param s3_client:
    :param bucket_name: The bucket that contains the object.
    :param key_name: The key of the object to be downloaded.
    :return: The object data in bytes.
```

```
    """
    try:
        response = s3_client.get_object(Bucket=bucket_name, Key=key_name)
        body = response['Body'].read()
        print(f"Got object '{key_name}' from bucket '{bucket_name}'.")
    except ClientError:
        print(f"Couldn't get object '{key_name}' from bucket '{bucket_name}'.")
        raise
    else:
        return body

def main():
    s3_client = boto3.client('s3')
    resp = get_object(s3_client, 'doc-example-bucket--use1-az4--x-s3', 'sample.txt')
    print(resp)

if __name__ == "__main__":
     main()
```

Using the AWS CLI

The following `get-object` example command shows how you can use the AWS CLI to download an object from Amazon S3. This command gets the object *KEY_NAME* from the directory bucket *bucket-base-name*`--`*zone-id*`--x-s3`. The object will be downloaded to a file named *LOCAL_FILE_NAME*. To run this command, replace the *user input placeholders* with your own information.

```
aws s3api get-object --bucket bucket-base-name--zone-id--x-s3 --
key KEY_NAME LOCAL_FILE_NAME
```

For more information, see get-object in the *AWS CLI Command Reference*.

Generating presigned URLs to share objects directory bucket

The following code examples show how to generate presigned URLs to share objects from an Amazon S3 directory bucket.

Using the AWS CLI

The following example command shows how you can use the AWS CLI to generate a presigned URL for an object from Amazon S3. This command generates a presigned URL for an object *KEY_NAME*

from the directory bucket *bucket-base-name*--*zone-id*--x-s3. To run this command, replace the *user input placeholders* with your own information.

```
aws s3 presign s3://bucket-base-name--zone-id--x-s3/KEY_NAME --expires-in 7200
```

For more information, see presign in the *AWS CLI Command Reference*.

Retrieving object metadata from directory buckets

The following AWS SDK and AWS CLI examples show how to use the `HeadObject` and `GetObjectAttributes` API operation to retrieve metadata from an object in an Amazon S3 directory bucket without returning the object itself.

Using the AWS SDKs

SDK for Java 2.x

Example

```
public static void headObject(S3Client s3Client, String bucketName, String
 objectKey) {
    try {
        HeadObjectRequest headObjectRequest = HeadObjectRequest
                .builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();
        HeadObjectResponse response = s3Client.headObject(headObjectRequest);
        System.out.format("Amazon S3 object: \"%s\" found in bucket: \"%s\" with
 ETag: \"%s\"", objectKey, bucketName, response.eTag());
    }
    catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
```

Using the AWS CLI

The following `head-object` example command shows how you can use the AWS CLI to retrieve metadata from an object. To run this command, replace the *user input placeholders* with your own information.

```
aws s3api head-object --bucket bucket-base-name--zone-id--x-s3 --key KEY_NAME
```

For more information, see head-object in the *AWS CLI Command Reference*.

The following `get-object-attributes` example command shows how you can use the AWS CLI to retrieve metadata from an object. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api get-object-attributes --bucket bucket-base-name--zone-id--x-s3 --key KEY_NAME
 --object-attributes "StorageClass" "ETag" "ObjectSize"
```

For more information, see get-object-attributes in the *AWS CLI Command Reference*.

Listing objects from a directory bucket

The following code examples show how to list objects in an Amazon S3 directory bucket by using the `ListObjectsV2` API operation.

Using the AWS CLI

The following `list-objects-v2` example command shows how you can use the AWS CLI to list objects from Amazon S3. This command lists objects from the directory bucket *`bucket-base-name--zone-id`*`--x-s3`. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api list-objects-v2 --bucket bucket-base-name--zone-id--x-s3
```

For more information, see list-objects-v2 in the *AWS CLI Command Reference*.

Security for directory buckets

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from data centers and network architectures that are built to meet the requirements of the most security-sensitive organizations. Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security of the cloud and security in the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-

party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs.

To learn about the compliance programs, see AWS services in Scope by Compliance Program.

- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors, including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation will help you understand how to apply the shared responsibility model when using directory buckets. The following topics show you how to configure directory buckets to meet your security and compliance objectives. You will also learn how to use other AWS services that can help you monitor and secure your objects in directory buckets.

Data protection and encryption

For more information about how you can configure encryption for directory buckets, see the following topics.

Topics

- Server-side encryption
- Setting and monitoring default encryption for directory buckets
- Using server-side encryption with AWS KMS keys (SSE-KMS) in directory buckets
- Encryption in transit
- Data deletion

Server-side encryption

All directory buckets have encryption configured by default, and all new objects that are uploaded to directory buckets are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every directory bucket. If you want to specify a different encryption type, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), by setting the default encryption configuration of the bucket. For more information about SSE-KMS in directory buckets, see Using server-side encryption with AWS KMS keys (SSE-KMS) in directory buckets.

We recommend that the bucket's default encryption uses the desired encryption configuration and you don't override the bucket default encryption in your `CreateSession` requests or PUT object

requests. Then, new objects are automatically encrypted with the desired encryption settings. For more information about the encryption overriding behaviors in directory buckets, see Specifying server-side encryption with AWS KMS for new object uploads.

SSE-KMS with directory buckets differs from SSE-KMS in general purpose buckets in the following aspects.

- Your SSE-KMS configuration can only support 1 customer managed key per directory bucket for the lifetime of the bucket. The AWS managed key (`aws/s3`) isn't supported. Also, after you specify a customer managed key for SSE-KMS, you can't override the customer managed key for the bucket's SSE-KMS configuration.

  You can identify the customer managed key you specified for the bucket's SSE-KMS configuration, in the following way:

  - You make a `HeadObject` API operation request to find the value of `x-amz-server-side-encryption-aws-kms-key-id` in your response.

  To use a new customer managed key for your data, we recommend copying your existing objects to a new directory bucket with a new customer managed key.
- For Zonal endpoint (object-level) API operations except CopyObject and UploadPartCopy, you authenticate and authorize requests through CreateSession for low latency. We recommend that the bucket's default encryption uses the desired encryption configuration and you don't override the bucket default encryption in your `CreateSession` requests or PUT object requests. Then, new objects are automatically encrypted with the desired encryption settings. To encrypt new objects in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default encryption configuration with a KMS key (specifically, a customer managed key). Then, when a session is created for Zonal endpoint API operations, new objects are automatically encrypted and decrypted with SSE-KMS and S3 Bucket Keys during the session. For more information about the encryption overriding behaviors in directory buckets, see Specifying server-side encryption with AWS KMS for new object uploads.

  In the Zonal endpoint API calls (except CopyObject and UploadPartCopy), you can't override the values of the encryption settings (`x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id`, `x-amz-server-side-encryption-context`, and `x-amz-server-side-encryption-bucket-key-enabled`) from the `CreateSession` request. You don't need to explicitly specify these encryption settings values in Zonal endpoint API calls, and Amazon S3 will use the encryption settings values from the `CreateSession` request to protect new objects in the directory bucket.

> **Note**
>
> When you use the AWS CLI or the AWS SDKs, for `CreateSession`, the session token refreshes automatically to avoid service interruptions when a session expires. The AWS CLI or the AWS SDKs use the bucket's default encryption configuration for the `CreateSession` request. It's not supported to override the encryption settings values in the `CreateSession` request. Also, in the Zonal endpoint API calls (except CopyObject and UploadPartCopy), it's not supported override the values of the encryption settings from the `CreateSession` request.

- For CopyObject, to encrypt new object copies in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default encryption configuration with a KMS key (specifically, a customer managed key). Then, when you specify server-side encryption settings for new object copies with SSE-KMS, you must make sure the encryption key is the same customer managed key that you specified for the directory bucket's default encryption configuration. For UploadPartCopy, to encrypt new object part copies in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default encryption configuration with a KMS key (specifically, a customer managed key). You can't specify server-side encryption settings for new object part copies with SSE-KMS in the UploadPartCopy request headers. Also, the encryption settings that you provide in the CreateMultipartUpload request must match the default encryption configuration of the destination bucket.
- S3 Bucket Keys are always enabled for GET and PUT operations in a directory bucket and can't be disabled. S3 Bucket Keys aren't supported, when you copy SSE-KMS encrypted objects from general purpose buckets to directory buckets, from directory buckets to general purpose buckets, or between directory buckets, through CopyObject, UploadPartCopy, the Copy operation in Batch Operations, or the import jobs. In this case, Amazon S3 makes a call to AWS KMS every time a copy request is made for a KMS-encrypted object.
- When you specify an AWS KMS customer managed key for encryption in your directory bucket, only use the key ID or key ARN. The key alias format of the KMS key isn't supported.

Directory buckets don't support dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS), or server-side encryption with customer-provided encryption keys (SSE-C).

Setting and monitoring default encryption for directory buckets

Amazon S3 buckets have bucket encryption enabled by default, and new objects are automatically encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3). This encryption applies to all new objects in your Amazon S3 buckets, and comes at no cost to you.

If you need more control over your encryption keys, such as managing key rotation and access policy grants, you can elect to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS).

Note

- We recommend that the bucket's default encryption uses the desired encryption configuration and you don't override the bucket default encryption in your `CreateSession` requests or PUT object requests. Then, new objects are automatically encrypted with the desired encryption settings. For more information about the encryption overriding behaviors in directory buckets, see Specifying server-side encryption with AWS KMS for new object uploads.
- To encrypt new objects in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default encryption configuration with a KMS key (specifically, a customer managed key). Then, when a session is created for Zonal endpoint API operations, new objects are automatically encrypted and decrypted with SSE-KMS and S3 Bucket Keys during the session.
- When you set default bucket encryption to SSE-KMS, S3 Bucket Keys are always enabled for GET and PUT operations in a directory bucket and can't be disabled. S3 Bucket Keys aren't supported, when you copy SSE-KMS encrypted objects from general purpose buckets to directory buckets, from directory buckets to general purpose buckets, or between directory buckets, through CopyObject, UploadPartCopy, the Copy operation in Batch Operations, or the import jobs. In this case, Amazon S3 makes a call to AWS KMS every time a copy request is made for a KMS-encrypted object. For more information about how S3 Bucket Keys reduce your AWS KMS request costs, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.
- When you specify an AWS KMS customer managed key for encryption in your directory bucket, only use the key ID or key ARN. The key alias format of the KMS key isn't supported.

- Dual-layer server-side encryption with AWS KMS keys (DSSE-KMS) and server-side encryption with customer-provided keys (SSE-C) aren't supported for default encryption in directory buckets.

For more information about configuring default encryption, see Configuring default encryption.

For more information about the permissions required for default encryption, see PutBucketEncryption in the *Amazon Simple Storage Service API Reference*.

You can configure Amazon S3 default encryption for an S3 bucket by using the Amazon S3 console, the AWS SDKs, the Amazon S3 REST API, and the AWS Command Line Interface (AWS CLI).

Using the S3 console

To configure default encryption on an Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want.
4. Choose the **Properties** tab.
5. Under **Server-side encryption settings**, directory buckets use Server-side encryption with **Amazon S3 managed keys (SSE-S3)**.
6. Choose **Save changes**.

Using the AWS CLI

These examples show you how to configure default encryption by using SSE-S3 or by using SSE-KMS with an S3 Bucket Key.

For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets. For more information about using the AWS CLI to configure default encryption, see put-bucket-encryption.

Example– Default encryption with SSE-S3

This example configures default bucket encryption with Amazon S3 managed keys. To use the command, replace the *`user input placeholders`* with your own information.

```
aws s3api put-bucket-encryption --bucket bucket-base-name--zone-id--x-s3 --server-side-
encryption-configuration '{
    "Rules": [
        {
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }
    ]
}'
```

Example– Default encryption with SSE-KMS using an S3 Bucket Key

This example configures default bucket encryption with SSE-KMS using an S3 Bucket Key. To use the command, replace the *user input placeholders* with your own information.

```
aws s3api put-bucket-encryption --bucket bucket-base-name--zone-id--x-s3 --server-side-
encryption-configuration '{
    "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

Using the REST API

Use the REST API `PutBucketEncryption` operation to set default encryption with a type of server-side encryption to use — SSE-S3, or SSE-KMS.

For more information, see PutBucketEncryption in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

When using AWS SDKs, you can request Amazon S3 to use AWS KMS keys for server-side encryption. The following AWS SDKs for Java and .NET examples configure default encryption

configuration for a directory bucket with SSE-KMS and an S3 Bucket Key. For information about other SDKs, see Sample code and libraries on the AWS Developer Center.

> **Important**
>
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

Java

With the AWS SDK for Java 2.x, you can request Amazon S3 to use an AWS KMS key by using the `applyServerSideEncryptionByDefault` method to specify the default encryption configuration of your directory bucket for data encryption with SSE-KMS. You create a symmetric encryption KMS key and specify that in the request.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

public class Main {
    public static void main(String[] args) {
        S3Client s3 = S3Client.create();
        String bucketName = "bucket-base-name--zoneid--x-s3";
        String kmsKeyId = "your-kms-customer-managed-key-id";

        // AWS managed KMS keys aren't supported. Only customer-managed keys are
 supported.
        ServerSideEncryptionByDefault serverSideEncryptionByDefault =
 ServerSideEncryptionByDefault.builder()
                .sseAlgorithm(ServerSideEncryption.AWS_KMS)
                .kmsMasterKeyID(kmsKeyId)
                .build();

        // The bucketKeyEnabled field is enforced to be true.
```

```
        ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
                .bucketKeyEnabled(true)
                .applyServerSideEncryptionByDefault(serverSideEncryptionByDefault)
                .build();

        ServerSideEncryptionConfiguration serverSideEncryptionConfiguration =
 ServerSideEncryptionConfiguration.builder()
                .rules(rule)
                .build();

        PutBucketEncryptionRequest putRequest = PutBucketEncryptionRequest.builder()
                .bucket(bucketName)

 .serverSideEncryptionConfiguration(serverSideEncryptionConfiguration)
                .build();

        s3.putBucketEncryption(putRequest);

    }
}
```

For more information about creating customer managed keys, see Programming the AWS KMS API in the *AWS Key Management Service Developer Guide*.

For working code examples of uploading an object, see the following topics. To use these examples, you must update the code examples and provide encryption information as shown in the preceding code fragment.

- For uploading an object in a single operation, see Uploading objects to a directory bucket.
- For multipart upload API operations, see Using multipart uploads with directory buckets.

.NET

With the AWS SDK for .NET, you can request Amazon S3 to use an AWS KMS key by using the `ServerSideEncryptionByDefault` property to specify the default encryption configuration of your directory bucket for data encryption with SSE-KMS. You create a symmetric encryption customer managed key and specify that in the request.

```
    // Set the bucket server side encryption to use AWSKMS with a customer-managed
 key id.
    // bucketName: Name of the directory bucket. "bucket-base-name--zonsid--x-s3"
    // kmsKeyId: The Id of the customer managed KMS Key. "your-kms-customer-managed-
key-id"
    // Returns True if successful.
    public static async Task<bool> SetBucketServerSideEncryption(string bucketName,
 string kmsKeyId)
    {
        var serverSideEncryptionByDefault = new ServerSideEncryptionConfiguration
        {
            ServerSideEncryptionRules = new List<ServerSideEncryptionRule>
            {
                new ServerSideEncryptionRule
                {
                    ServerSideEncryptionByDefault = new
 ServerSideEncryptionByDefault
                    {
                        ServerSideEncryptionAlgorithm =
 ServerSideEncryptionMethod.AWSKMS,
                        ServerSideEncryptionKeyManagementServiceKeyId = kmsKeyId
                    }
                }
            }
        };
        try
        {
            var encryptionResponse =await _s3Client.PutBucketEncryptionAsync(new
 PutBucketEncryptionRequest
            {
                BucketName = bucketName,
                ServerSideEncryptionConfiguration = serverSideEncryptionByDefault,
            });

            return encryptionResponse.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "AccessDenied"
                ? $"This account does not have permission to set encryption on
 {bucketName}, please try again."
                : $"Unable to set bucket encryption for bucket {bucketName},
 {ex.Message}");
        }
```

```
        return false;
    }
```

For more information about creating customer managed keys, see Programming the AWS KMS API in the *AWS Key Management Service Developer Guide*.

For working code examples of uploading an object, see the following topics. To use these examples, you must update the code examples and provide encryption information as shown in the preceding code fragment.

- For uploading an object in a single operation, see Uploading objects to a directory bucket.
- For multipart upload API operations, see Using multipart uploads with directory buckets.

Monitoring default encryption for directory buckets with AWS CloudTrail

You can track default encryption configuration requests for Amazon S3 directory buckets by using AWS CloudTrail events. The following API event names are used in CloudTrail logs:

- `PutBucketEncryption`
- `GetBucketEncryption`
- `DeleteBucketEncryption`

Note

- EventBridge isn't supported in directory buckets.
- Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS) or server-side encryption with customer-provided encryption keys (SSE-C) aren't supported in directory buckets.

For more information about monitoring default encryption with AWS CloudTrail, see Monitoring default encryption with AWS CloudTrail and Amazon EventBridge.

Using server-side encryption with AWS KMS keys (SSE-KMS) in directory buckets

The security controls in AWS KMS can help you meet encryption-related compliance requirements. You can choose to configure directory buckets to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) and use these KMS keys to protect your data in Amazon S3 directory buckets. For more information about SSE-KMS, see Using server-side encryption with AWS KMS keys (SSE-KMS).

Permissions

To upload or download an object encrypted with an AWS KMS key to or from Amazon S3, you need `kms:GenerateDataKey` and `kms:Decrypt` permissions on the key. For more information, see Allow key users to use a KMS key for cryptographic operations in the *AWS Key Management Service Developer Guide*. For information about the AWS KMS permissions that are required for multipart uploads, see Multipart upload API and permissions.

For more information about KMS keys for SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS).

Topics

- AWS KMS keys
- Using SSE-KMS for cross-account operations
- Amazon S3 Bucket Keys
- Requiring SSE-KMS
- Encryption context
- Sending requests for AWS KMS encrypted objects
- Auditing SSE-KMS encryption in directory buckets
- Specifying server-side encryption with AWS KMS (SSE-KMS) for new object uploads in directory buckets

AWS KMS keys

Your SSE-KMS configuration can only support 1 customer managed key per directory bucket for the lifetime of the bucket. The AWS managed key (`aws/s3`) isn't supported. Also, after you specify a customer managed key for SSE-KMS, you can't override the customer managed key for the bucket's SSE-KMS configuration.

You can identify the customer managed key you specified for the bucket's SSE-KMS configuration, in the following way:

- You make a `HeadObject` API operation request to find the value of `x-amz-server-side-encryption-aws-kms-key-id` in your response.

To use a new customer managed key for your data, we recommend copying your existing objects to a new directory bucket with a new customer managed key.

When you specify an AWS KMS customer managed key for encryption in your directory bucket, only use the key ID or key ARN. The key alias format of the KMS key isn't supported.

For more information about KMS keys for SSE-KMS, see AWS KMS keys.

Using SSE-KMS for cross-account operations

When using encryption for cross-account operations in directory buckets, be aware of the following:

- If you want to grant cross-account access to your S3 objects, configure a policy of a customer managed key to allow access from another account.
- To specify a customer managed key, you must use a fully qualified KMS key ARN.

Amazon S3 Bucket Keys

S3 Bucket Keys are always enabled for `GET` and `PUT` operations in a directory bucket and can't be disabled. S3 Bucket Keys aren't supported, when you copy SSE-KMS encrypted objects from general purpose buckets to directory buckets, from directory buckets to general purpose buckets, or between directory buckets, through CopyObject, UploadPartCopy, the Copy operation in Batch Operations, or the import jobs. In this case, Amazon S3 makes a call to AWS KMS every time a copy request is made for a KMS-encrypted object.

For Zonal endpoint (object-level) API operations except CopyObject and UploadPartCopy, you authenticate and authorize requests through CreateSession for low latency. We recommend that the bucket's default encryption uses the desired encryption configuration and you don't override the bucket default encryption in your `CreateSession` requests or `PUT` object requests. Then, new objects are automatically encrypted with the desired encryption settings. To encrypt new objects in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default

encryption configuration with an KMS key (specifically, a customer managed key). Then, when a session is created for Zonal endpoint API operations, new objects are automatically encrypted and decrypted with SSE-KMS and S3 Bucket Keys during the session. For more information about the encryption overriding behaviors in directory buckets, see Specifying server-side encryption with AWS KMS for new object uploads.

S3 Bucket Keys are used for a time-limited period within Amazon S3, further reducing the need for Amazon S3 to make requests to AWS KMS to complete encryption operations. For more information about using S3 Bucket Keys, see Amazon S3 Bucket Keys and Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

Requiring SSE-KMS

To require SSE-KMS of all objects in a particular directory bucket, you can use a bucket policy. For example, when you use the `CreateSession` API operation to grant permission to upload a new object (`PutObject`, `CopyObject`, and `CreateMultipartUpload`), the following bucket policy denies the upload object permission (`s3express:CreateSession`) to everyone if the `CreateSession` request doesn't include an `x-amz-server-side-encryption-aws-kms-key-id` header that requests SSE-KMS.

JSON

```
{
   "Version":"2012-10-17",
   "Id":"UploadObjectPolicy",
   "Statement":[{
         "Sid":"DenyObjectsThatAreNotSSEKMS",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3express:CreateSession",
         "Resource":"arn:aws:s3express:us-east-1:111122223333:bucket/amzn-s3-
demo-bucket--usw2-az1--x-s3",
         "Condition":{
            "Null":{
               "s3express:x-amz-server-side-encryption-aws-kms-key-id":"true"
            }
         }
      }
   ]
}
```

To require that a particular AWS KMS key be used to encrypt the objects in a bucket, you can use the `s3express:x-amz-server-side-encryption-aws-kms-key-id` condition key. To specify the KMS key, you must use a key Amazon Resource Name (ARN) that is in the `arn:aws:kms:region:acct-id:key/key-id` format. AWS Identity and Access Management does not validate if the string for `s3express:x-amz-server-side-encryption-aws-kms-key-id` exists. The AWS KMS key ID that Amazon S3 uses for object encryption must match the AWS KMS key ID in the policy, otherwise Amazon S3 denies the request.

For more information about how to use SSE-KMS for new object uploads, see Specifying server-side encryption with AWS KMS (SSE-KMS) for new object uploads in directory buckets.

For a complete list of specific condition keys for directory buckets, see Authorizing Regional endpoint API operations with IAM.

Encryption context

For directory buckets, an *encryption context* is a set of key-value pairs that contains contextual information about the data. An additional encryption context value is not supported. For more information about the encryption context, see Encryption context.

By default, if you use SSE-KMS on a directory bucket, Amazon S3 uses the bucket Amazon Resource Name (ARN) as the encryption context pair:

```
arn:aws:s3express:region:account-id:bucket/bucket-base-name--zone-id--x-s3
```

Make sure your IAM policies or AWS KMS key policies use your bucket ARN as the encryption context.

You can optionally provide an explicit encryption context pair by using the `x-amz-server-side-encryption-context` header in a Zonal endpoint API request, such as CreateSession. The value of this header is a Base64-encoded string of a UTF-8 encoded JSON, which contains the encryption context as key-value pairs. For directory buckets, the encryption context must match the default encryption context – the bucket Amazon Resource Name (ARN). Also, because the encryption context is not encrypted, make sure it does not include sensitive information.

You can use the encryption context to identify and categorize your cryptographic operations. You can also use the default encryption context ARN value to track relevant requests in AWS CloudTrail by viewing which directory bucket ARN was used with which encryption key.

In the `requestParameters` field of a CloudTrail log file, if you use SSE-KMS on a directory bucket, the encryption context value is the ARN of the bucket.

```
"encryptionContext": {
    "aws:s3express:arn": "arn:aws:s3:::arn:aws:s3express:region:account-id:bucket/bucket-base-name--zone-id--x-s3"
}
```

Also, for object encryption with SSE-KMS in a directory bucket, your AWS KMS CloudTrail events log your bucket ARN instead of your object ARN.

Sending requests for AWS KMS encrypted objects

Directory buckets can only be accessed through HTTPS (TLS). Also, directory buckets sign requests by using AWS Signature Version 4 (SigV4). For more information about sending requests for AWS KMS encrypted objects, see Sending requests for AWS KMS encrypted objects.

If your object uses SSE-KMS, don't send encryption request headers for GET requests and HEAD requests. Otherwise, you'll get an HTTP 400 Bad Request error.

Auditing SSE-KMS encryption in directory buckets

To audit the usage of your AWS KMS keys for your SSE-KMS encrypted data, you can use AWS CloudTrail logs. You can get insight into your cryptographic operations, such as GenerateDataKey and Decrypt. CloudTrail supports numerous attribute values for filtering your search, including event name, user name, and event source.

Topics

- Specifying server-side encryption with AWS KMS (SSE-KMS) for new object uploads in directory buckets

Specifying server-side encryption with AWS KMS (SSE-KMS) for new object uploads in directory buckets

For directory buckets, to encrypt your data with server-side encryption, you can use either server-side encryption with Amazon S3 managed keys (SSE-S3) (the default) or server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS). We recommend that the bucket's default encryption uses the desired encryption configuration and you don't override the bucket default encryption in your `CreateSession` requests or PUT object requests. Then, new objects are automatically encrypted with the desired encryption settings. For more information about the

encryption overriding behaviors in directory buckets, see Specifying server-side encryption with AWS KMS for new object uploads.

All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. If you want to specify a different encryption type for a directory bucket, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS). To encrypt new objects in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default encryption configuration with a KMS key (specifically, a customer managed key). The AWS managed key (`aws/s3`) isn't supported. Your SSE-KMS configuration can only support 1 customer managed key per directory bucket for the lifetime of the bucket. After you specify a customer managed key for SSE-KMS, you can't override the customer managed key for the bucket's SSE-KMS configuration. Then, when you specify server-side encryption settings for new objects with SSE-KMS, you must make sure the encryption key is the same customer managed key that you specified for the directory bucket's default encryption configuration. To use a new customer managed key for your data, we recommend copying your existing objects to a new directory bucket with a new customer managed key.

You can apply encryption when you are either uploading a new object or copying an existing object. If you change an object's encryption, a new object is created to replace the old one.

You can specify SSE-KMS by using the REST API operations, AWS SDKs, and the AWS Command Line Interface (AWS CLI).

Note

- For directory buckets, the encryption overriding behaviors are as follows:
  - When you use CreateSession with the REST API to authenticate and authorize Zonal endpoint API requests except CopyObject and UploadPartCopy, you can override the encryption settings to SSE-S3 or to SSE-KMS only if you specified the bucket’s default encryption with SSE-KMS previously.
  - When you use CreateSession with the AWS CLI or the AWS SDKs to authenticate and authorize Zonal endpoint API requests except CopyObject and UploadPartCopy, you can’t override the encryption settings at all.
  - When you make CopyObject requests, you can override the encryption settings to SSE-S3 or to SSE-KMS only if you specified the bucket’s default encryption with SSE-

KMS previously. When you make UploadPartCopy requests, you can't override the encryption settings.

- You can use multi-Region AWS KMS keys in Amazon S3. However, Amazon S3 currently treats multi-Region keys as though they were single-Region keys, and does not use the multi-Region features of the key. For more information, see Using multi-Region keys in *AWS Key Management Service Developer Guide*.
- If you want to use a KMS key that's owned by a different account, you must have permission to use the key. For more information about cross-account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*.

Using the REST API

Note

Only 1 customer managed key is supported per directory bucket for the lifetime of the bucket. The AWS managed key (`aws/s3`) isn't supported. After you specify SSE-KMS as your bucket's default encryption configuration with a customer managed key, you can't change the customer managed key for the bucket's SSE-KMS configuration.

For Zonal endpoint (object-level) API operations except CopyObject and UploadPartCopy, you authenticate and authorize requests through CreateSession for low latency. We recommend that the bucket's default encryption uses the desired encryption configurations and you don't override the bucket default encryption in your `CreateSession` requests or PUT object requests. Then, new objects are automatically encrypted with the desired encryption settings. To encrypt new objects in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default encryption configuration with a KMS key (specifically, a customer managed key). Then, when a session is created for Zonal endpoint API operations, new objects are automatically encrypted and decrypted with SSE-KMS and S3 Bucket Keys during the session. For more information about the encryption overriding behaviors in directory buckets, see Specifying server-side encryption with AWS KMS for new object uploads.

In the Zonal endpoint API calls (except CopyObject and UploadPartCopy) using the REST API, you can't override the values of the encryption settings (`x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id`, `x-amz-server-side-encryption-context`,

and `x-amz-server-side-encryption-bucket-key-enabled`) from the `CreateSession` request. You don't need to explicitly specify these encryption settings values in Zonal endpoint API calls, and Amazon S3 will use the encryption settings values from the `CreateSession` request to protect new objects in the directory bucket.

> **Note**
>
> When you use the AWS CLI or the AWS SDKs, for `CreateSession`, the session token refreshes automatically to avoid service interruptions when a session expires. The AWS CLI or the AWS SDKs use the bucket's default encryption configuration for the `CreateSession` request. It's not supported to override the encryption settings values in the `CreateSession` request. Also, in the Zonal endpoint API calls (except CopyObject and UploadPartCopy), it's not supported to override the values of the encryption settings from the `CreateSession` request.

For CopyObject, to encrypt new object copies in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default encryption configuration with a KMS key (specifically, a customer managed key). Then, when you specify server-side encryption settings for new object copies with SSE-KMS, you must make sure the encryption key is the same customer managed key that you specified for the directory bucket's default encryption configuration. For UploadPartCopy, to encrypt new object part copies in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default encryption configuration with a KMS key (specifically, a customer managed key). You can't specify server-side encryption settings for new object part copies with SSE-KMS in the UploadPartCopy request headers. Also, the encryption settings that you provide in the CreateMultipartUpload request must match the default encryption configuration of the destination bucket.

Topics

- Amazon S3 REST API operations that support SSE-KMS
- Encryption context (x-amz-server-side-encryption-context)
- AWS KMS key ID (x-amz-server-side-encryption-aws-kms-key-id)
- S3 Bucket Keys (x-amz-server-side-encryption-aws-bucket-key-enabled)

Amazon S3 REST API operations that support SSE-KMS

The following object-level REST API operations in directory buckets accept the `x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id`, and `x-amz-server-side-encryption-context` request headers.

- CreateSession – When you use Zonal endpoint (object-level) API operations (except CopyObject and UploadPartCopy), you can specify these request headers.
- PutObject – When you upload data by using the PUT API operation, you can specify these request headers.
- CopyObject – When you copy an object, you have both a source object and a target object. When you pass SSE-KMS headers with the `CopyObject` operation, they're applied only to the target object.
- CreateMultipartUpload – When you upload large objects by using the multipart upload API operation, you can specify these headers. You specify these headers in the `CreateMultipartUpload` request.

The response headers of the following REST API operations return the `x-amz-server-side-encryption` header when an object is stored by using server-side encryption.

- CreateSession
- PutObject
- CopyObject
- POST Object
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

Important

- All GET and PUT requests for an object protected by AWS KMS fail if you don't make these requests by using Transport Layer Security (TLS), or Signature Version 4.
- If your object uses SSE-KMS, don't send encryption request headers for GET requests and HEAD requests, or you'll get an HTTP 400 BadRequest error.

Encryption context (`x-amz-server-side-encryption-context`)

If you specify `x-amz-server-side-encryption:aws:kms`, the Amazon S3 API supports you to optionally provide an explicit encryption context with the `x-amz-server-side-encryption-context` header. For directory buckets, an encryption context is a set of key-value pairs that contain contextual information about the data. The value must match the default encryption context — the bucket Amazon Resource Name (ARN). An additional encryption context value is not supported.

For information about the encryption context in directory buckets, see Encryption context. For general information about the encryption context, see AWS Key Management Service Concepts - Encryption context in the *AWS Key Management Service Developer Guide*.

AWS KMS key ID (`x-amz-server-side-encryption-aws-kms-key-id`)

You can use the `x-amz-server-side-encryption-aws-kms-key-id` header to specify the ID of the customer managed key that's used to protect the data.

Your SSE-KMS configuration can only support 1 customer managed key per directory bucket for the lifetime of the bucket. The AWS managed key (`aws/s3`) isn't supported. Also, after you specify a customer managed key for SSE-KMS, you can't override the customer managed key for the bucket's SSE-KMS configuration.

You can identify the customer managed key you specified for the bucket's SSE-KMS configuration, in the following way:

- You make a `HeadObject` API operation request to find the value of `x-amz-server-side-encryption-aws-kms-key-id` in your response.

To use a new customer managed key for your data, we recommend copying your existing objects to a new directory bucket with a new customer managed key.

For information about the encryption context in directory buckets, see AWS KMS keys.

S3 Bucket Keys (`x-amz-server-side-encryption-aws-bucket-key-enabled`)

S3 Bucket Keys are always enabled for `GET` and `PUT` operations in a directory bucket and can't be disabled. S3 Bucket Keys aren't supported, when you copy SSE-KMS encrypted objects from general purpose buckets to directory buckets, from directory buckets to general purpose buckets, or between directory buckets, through CopyObject, UploadPartCopy, the Copy operation in Batch Operations, or the import jobs. In this case, Amazon S3 makes a call to AWS KMS every time a copy request is made for a KMS-encrypted object. For information about the S3 Bucket Keys in directory buckets, see Encryption context.

Using the AWS CLI

> **Note**
>
> When you use the AWS CLI, for `CreateSession`, the session token refreshes automatically to avoid service interruptions when a session expires. It's not supported to override the encryption settings values for the `CreateSession` request. Also, in the Zonal endpoint API calls (except CopyObject and UploadPartCopy), it's not supported to override the values of the encryption settings from the `CreateSession` request.
> To encrypt new objects in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default encryption configuration with a KMS key (specifically, a customer managed key). Then, when a session is created for Zonal endpoint API operations, new objects are automatically encrypted and decrypted with SSE-KMS and S3 Bucket Keys during the session.

To use the following example AWS CLI commands, replace the *`user input placeholders`* with your own information.

When you upload a new object or copy an existing object, you can specify the use of server-side encryption with AWS KMS keys to encrypt your data. To do this, use the `put-bucket-encryption` command to set the directory bucket's default encryption configuration as SSE-KMS (`aws:kms`). Specifically, add the `--server-side-encryption aws:kms` header to the request. Use the `--ssekms-key-id` *`example-key-id`* to add your customer managed AWS KMS key that you created. If you specify `--server-side-encryption aws:kms`, you must provide an AWS KMS key ID of your customer managed key. Directory buckets don't use an AWS managed key. For an example command, see Using the AWS CLI.

Then, when you upload a new object with the following command, Amazon S3 uses the bucket settings for default encryption to encrypt the object by default.

```
aws s3api put-object --bucket bucket-base-name--zone-id--x-s3 --key example-object-key
 --body filepath
```

You don't need to add `-\-bucket-key-enabled` explicitly in your Zonal endpoint API operations commands. S3 Bucket Keys are always enabled for `GET` and `PUT` operations in a directory bucket and can't be disabled. S3 Bucket Keys aren't supported, when you copy SSE-KMS encrypted objects from general purpose buckets to directory buckets, from directory buckets to general purpose buckets, or between directory buckets, through CopyObject, UploadPartCopy, the Copy operation in Batch Operations, or the import jobs. In this case, Amazon S3 makes a call to AWS KMS every time a copy request is made for a KMS-encrypted object.

You can copy an object from a source bucket (for example, a general purpose bucket) to a new bucket (for example, a directory bucket) and use SSE-KMS encryption for the destination objects. To do this, use the `put-bucket-encryption` command to set the default encryption configuration of the destination bucket (for example, a directory bucket) as SSE-KMS (`aws:kms`). For an example command, see Using the AWS CLI. Then, when you copy an object with the following command, Amazon S3 uses the bucket settings for default encryption to encrypt the object by default.

```
aws s3api copy-object --copy-source amzn-s3-demo-bucket/example-object-key --
bucket bucket-base-name--zone-id--x-s3 --key example-object-key
```

Using the AWS SDKs

When using AWS SDKs, you can request Amazon S3 to use AWS KMS keys for server-side encryption. The following examples show how to use SSE-KMS with the AWS SDKs for Java and .NET. For information about other SDKs, see Sample code and libraries on the AWS Developer Center.

> **Note**
>
> When you use the AWS SDKs, for `CreateSession`, the session token refreshes automatically to avoid service interruptions when a session expires. It's not supported to override the encryption settings values for the `CreateSession` request. Also, in the Zonal endpoint API calls (except CopyObject and UploadPartCopy), it's not supported to override the values of the encryption settings from the `CreateSession` request.

To encrypt new objects in a directory bucket with SSE-KMS, you must specify SSE-KMS as the directory bucket's default encryption configuration with a KMS key (specifically, a customer managed key). Then, when a session is created for Zonal endpoint API operations, new objects are automatically encrypted and decrypted with SSE-KMS and S3 Bucket Keys during the session.
For more information about using AWS SDKs to set the default encryption configuration of a directory bucket as SSE-KMS, see Using the AWS SDKs.

Important

When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

For more information about creating customer managed keys, see Programming the AWS KMS API in the *AWS Key Management Service Developer Guide*.

Encryption in transit

Directory buckets use Regional and Zonal API endpoints. Depending on the Amazon S3 API operation that you use, either a Regional or Zonal endpoint is required. You can access Zonal and Regional endpoints through a gateway virtual private cloud (VPC) endpoint. There is no additional charge for using gateway endpoints. To learn more about Regional and Zonal API endpoints, see Networking for directory buckets.

Data deletion

You can delete one or more objects directly from your directory buckets by using the Amazon S3 console, AWS SDKs, AWS Command Line Interface (AWS CLI), or Amazon S3 REST API. Because all objects in your directory buckets incur storage costs, we recommend deleting objects that you no longer need.

Deleting an object that's stored in a directory bucket also recursively deletes any parent directories, if those parent directories don't contain any objects other than the object that's being deleted.

> **Note**
>
> Multi-factor authentication (MFA) delete and S3 Versioning are not supported for S3 Express One Zone.

Authenticating and authorizing requests

By default, directory buckets are private and can be accessed only by users who are explicitly granted access. The access control boundary for directory buckets is set only at the bucket level. In contrast, the access control boundary for general purpose buckets can be set at the bucket, prefix, or object tag level. This difference means that directory buckets are the only resource that you can include in bucket policies or IAM identity policies for S3 Express One Zone access.

Amazon S3 Express One Zone supports both AWS Identity and Access Management (AWS IAM) authorization and session-based authorization:

- To use Regional endpoint API operations (bucket-level, or control plane, operations) with S3 Express One Zone, you use the IAM authorization model, which doesn't involve session management. Permissions are granted for actions individually. For more information, see Authorizing Regional endpoint API operations with IAM.
- To use Zonal endpoint API operations (object-level, or data plane, operations), except for `CopyObject` and `HeadBucket`, you use the `CreateSession` API operation to create and manage sessions that are optimized for low-latency authorization of data requests. To retrieve and use a session token, you must allow the `s3express:CreateSession` action for your directory bucket in an identity-based policy or a bucket policy. For more information, see Authorizing Regional endpoint API operations with IAM. If you're accessing S3 Express One Zone in the Amazon S3 console, through the AWS Command Line Interface (AWS CLI), or by using the AWS SDKs, S3 Express One Zone creates a session on your behalf.

With the `CreateSession` API operation, you authenticate and authorize requests through a new session-based mechanism. You can use `CreateSession` to request temporary credentials that provide low-latency access to your bucket. These temporary credentials are scoped to a specific directory bucket.

To work with `CreateSession`, we recommend using the latest version of the AWS SDKs or using the AWS Command Line Interface (AWS CLI). The supported AWS SDKs and the AWS CLI handle session establishment, refreshment, and termination on your behalf.

You use session tokens with only Zonal (object-level) operations (except for `CopyObject` and `HeadBucket`) to distribute the latency that's associated with authorization over a number of requests in a session. For Regional endpoint API operations (bucket-level operations), you use IAM authorization, which doesn't involve managing a session. For more information, see Authorizing Regional endpoint API operations with IAM and Authorizing Zonal endpoint API operations with `CreateSession`.

How API operations are authenticated and authorized

The following table lists authentication and authorization information for directory bucket API operations. For each API operation, the table shows the API operation name, IAM policy action, endpoint type (Regional or Zonal), and authorization mechanism (IAM or session-based). This table also indicates whether cross-account access is supported. Access to bucket-level actions can be granted only in IAM identity-based policies (user or role), not bucket policies.

API	Endpoint type	IAM action	Cross-account access
`CreateBucket`	Regional	`s3express:CreateBucket`	No
`DeleteBucket`	Regional	`s3express:DeleteBucket`	No
`DeleteBucketInventoryConfiguration`	Regional	`s3express:PutInventoryConfiguration`	No
`DeleteBucketPolicy`	Regional	`s3express:DeleteBucketPolicy`	No
`GetBucketInventoryConfiguration`	Regional	`s3express:GetInventoryConfiguration`	No

API	Endpoint type	IAM action	Cross-account access
GetBucketPolicy	Regional	s3express:GetBucketPolicy	No
ListBucketInventoryConfigurations	Regional	s3express:GetInventoryConfiguration	No
ListDirectoryBuckets	Regional	s3express:ListAllMyDirectoryBuckets	No
PutBucketInventoryConfiguration	Regional	s3express:PutInventoryConfiguration	No
PutBucketPolicy	Regional	s3express:PutBucketPolicy	No
CreateSession	Zonal	s3express:CreateSession	Yes
CopyObject	Zonal	s3express:CreateSession	Yes
DeleteObject	Zonal	s3express:CreateSession	Yes
DeleteObjects	Zonal	s3express:CreateSession	Yes
HeadObject	Zonal	s3express:CreateSession	Yes
PutObject	Zonal	s3express:CreateSession	Yes
RenameObject	Zonal	s3express:CreateSession	No
GetObjectAttributes	Zonal	s3express:CreateSession	Yes
ListObjectsV2	Zonal	s3express:CreateSession	Yes

API	Endpoint type	IAM action	Cross-account access
HeadBucket	Zonal	s3express:CreateSession	Yes
CreateMultipartUpload	Zonal	s3express:CreateSession	Yes
UploadPart	Zonal	s3express:CreateSession	Yes
UploadPartCopy	Zonal	s3express:CreateSession	Yes
CompleteMultipartUpload	Zonal	s3express:CreateSession	Yes
AbortMultipartUpload	Zonal	s3express:CreateSession	Yes
ListParts	Zonal	s3express:CreateSession	Yes
ListMultipartUploads	Zonal	s3express:CreateSession	Yes
ListAccessPointsForDirectoryBuckets	Zonal	s3express:ListAccessPointsForDirectoryBuckets	Yes
GetAccessPointScope	Zonal	s3express:GetAccessPointScope	Yes
PutAccessPointScope	Zonal	s3express:PutAccessPointScope	Yes
DeleteAccessPointScope	Zonal	s3express:DeleteAccessPointScope	Yes

Topics

- Authorizing Regional endpoint API operations with IAM
- Authorizing Zonal endpoint API operations with CreateSession

Authorizing Regional endpoint API operations with IAM

AWS Identity and Access Management (IAM) is an AWS service that helps administrators securely control access to AWS resources. IAM administrators control who can be authenticated (signed in) and authorized (have permissions) to use Amazon S3 resources in directory buckets and S3 Express One Zone operations. You can use IAM for no additional charge.

By default, users don't have permissions for directory buckets. To grant access permissions for directory buckets, you can use IAM to create users, groups, or roles and attach permissions to those identities. For more information about IAM, see Security best practices in IAM in the *IAM User Guide*.

To provide access, you can add permissions to your users, groups, or roles through the following means:

- **Users and groups in AWS IAM Identity Center** – Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- **Users managed in IAM through an identity provider** – Create a role for identity federation. Follow the instructions in Creating a role for a third-party identity provider (federation) in the *IAM User Guide*.
- **IAM roles and users** – Create a role that your user can assume. Follow the instructions in Creating a role to delegate permissions to an IAM user in the *IAM User Guide*.

For more information about IAM for S3 Express One Zone, see the following topics.

Topics

- Principals
- Resources
- Actions for directory buckets
- IAM identity-based policies for directory buckets
- Example bucket policies for directory buckets
- AWS managed policies for Amazon S3 Express One Zone

Principals

When you create a resource-based policy to grant access to your buckets, you must use the `Principal` element to specify the person or application that can make a request for an action or operation on that resource. For directory bucket policies, you can use the following principals:

- An AWS account
- An IAM user
- An IAM role
- A federated user

For more information, see Principal in the *IAM User Guide*.

Resources

Amazon Resource Names (ARNs) for directory buckets contain the `s3express` namespace, the AWS Region, the AWS account ID, and the directory bucket name, which includes the AWS Zone ID. (an Availability Zone or Local Zone ID).

To access and perform actions on your directory bucket, you must use the following ARN format:

```
arn:aws:s3express:region:account-id:bucket/base-bucket-name--zone-id--x-s3
```

To access and perform actions on your access point for a directory bucket, you must use the following ARN format:

```
arn:aws::s3express:region:account-id:accesspoint/accesspoint-basename--zone-id--xa-s3
```

For more information about ARNs, see Amazon Resource Names (ARNs) in the *IAM User Guide*. For more information about resources, see IAM JSON Policy Elements: Resource in the *IAM User Guide*.

Actions for directory buckets

In an IAM identity-based policy or resource-based policy, you define which S3 actions are allowed or denied. Actions correspond to specific API operations. With directory buckets, you must use the S3 Express One Zone namespace to grant permissions, called `s3express`.

When you allow the `s3express:CreateSession` permission, the `CreateSession` API operation retrieves a temporary session token for all Zonal endpoint API (object level) operations. The session token returns credentials that are used for all other Zonal endpoint API operations. As a result, you

don't grant access permissions to Zonal API operations with IAM policies. Instead, `CreateSession` enables access for all object level operations. For the list of Zonal API operations and permissions, see Authenticating and authorizing requests.

To learn more about the `CreateSession` API operation, see CreateSession in the *Amazon Simple Storage Service API Reference*.

You can specify the following actions in the `Action` element of an IAM policy statement. Use policies to grant permissions to perform an operation in AWS. When you use an action in a policy, you usually allow or deny access to the API operation with the same name. However, in some cases, a single action controls access to more than one API operation. Access to bucket-level actions can be granted in only IAM identity-based policies (user or role) and not bucket policies.

For more information about how to configure access point policies, see Configuring IAM policies for using access points for directory buckets.

For more information, see Actions, resources, and condition keys for Amazon S3 Express.

IAM identity-based policies for directory buckets

Before you can create directory buckets, you must grant the necessary permissions to your AWS Identity and Access Management (IAM) role or users. This example policy allows access to the `CreateSession` API operation (for use with Zonal endpoint [object level] API operations) and all of the Regional endpoint (bucket-level) API operations. This policy allows the `CreateSession` API operation for use with all directory buckets, but the Regional endpoint API operations are allowed only for use with the specified directory bucket. To use this example policy, replace the *`user input placeholders`* with your own information.

> **Note**
>
> As a best practice, grant only the permissions required to perform a task (least-privilege). Remove any actions from this policy that are not needed for your use case. For a complete list of S3 Express One Zone actions, see Actions, resources, and condition keys for S3 Express One Zone in the *Service Authorization Reference*.

Example— Identity-based policy for directory bucket access

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "AllowRegionalEndpointAPIs",
            "Effect": "Allow",
            "Action": [
                "s3express:CreateBucket",
                "s3express:DeleteBucket",
                "s3express:DeleteBucketPolicy",
                "s3express:GetBucketPolicy",
                "s3express:PutBucketPolicy",
                "s3express:GetEncryptionConfiguration",
                "s3express:PutEncryptionConfiguration",
                "s3express:GetLifecycleConfiguration",
                "s3express:PutLifecycleConfiguration",
                "s3express:GetInventoryConfiguration",
                "s3express:PutInventoryConfiguration",
                "s3express:GetMetricsConfiguration",
                "s3express:PutMetricsConfiguration"
            ],
            "Resource": "arn:aws:s3express:region:account-id:bucket/bucket-base-name--
zone-id--x-s3"
        },
        {
            "Sid": "AllowListAndCreateSession",
            "Effect": "Allow",
            "Action": [
                "s3express:ListAllMyDirectoryBuckets",
                "s3express:CreateSession"
            ],
            "Resource": "*"
        }
    ]
}
```

This policy has two statements:

- The first statement grants permissions for Regional endpoint (bucket-level) API operations on a specific directory bucket. You can remove actions that you don't need for your use case.
- The second statement grants permissions for `ListAllMyDirectoryBuckets` and `CreateSession`. These actions don't support resource-level permissions, so the `Resource` is `"*"`. The `CreateSession` permission enables all Zonal endpoint (object-level) API operations, such as `PutObject`, `GetObject`, and `DeleteObject`.

Example bucket policies for directory buckets

This section provides example directory bucket policies. To use these policies, replace the *`user input placeholders`* with your own information.

The following example bucket policy allows AWS account ID *`111122223333`* to use the `CreateSession` API operation for the specified directory bucket. When no session mode is specified, the session will be created with the maximum allowable privilege (attempting `ReadWrite` first, then `ReadOnly` if not permitted). This policy grants access to the Zonal endpoint (object level) API operations.

Example– Bucket policy to allow CreateSession calls

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteAccess",
            "Effect": "Allow",
            "Resource": "arn:aws:s3express:us-west-2:111122223333:bucket/amzn-s3-
demo-bucket--usw2-az1--x-s3",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": [
                "s3express:CreateSession"
            ]
        }
    ]
}
```

Example– Bucket policy to allow CreateSession calls with a ReadOnly session

The following example bucket policy allows AWS account ID *`111122223333`* to use the `CreateSession` API operation. This policy uses the `s3express:SessionMode` condition key with the `ReadOnly` value to set a read-only session.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadOnlyAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": "111122223333"
            },
            "Action": "s3express:CreateSession",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3express:SessionMode": "ReadOnly"
                }
            }
        }
    ]
}
```

Example– Bucket policy to allow cross-account access for CreateSession calls

The following example bucket policy allows AWS account ID *111122223333* to use the `CreateSession` API operation for the specified directory bucket that's owned by AWS account ID *444455556666*.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccount",
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "s3express:CreateSession"
            ],
            "Resource": "arn:aws:s3express:us-west-2:444455556666:bucket/amzn-s3-
demo-bucket--usw2-az1--x-s3"
        }
    ]
}
```

AWS managed policies for Amazon S3 Express One Zone

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AmazonS3ExpressFullAccess

You can attach the `AmazonS3ExpressFullAccess` policy to your IAM identities. This policy grants full access to Amazon S3 Express One Zone directory buckets and operations. It allows all actions under the `s3express` service prefix on all resources.

This policy is intended for users or roles that need unrestricted access to directory buckets. This policy covers only Amazon S3 Express One Zone operations. For standard Amazon S3 operations, you need additional policies.

To view the permissions for this policy, see AmazonS3ExpressFullAccess in the AWS Managed Policy Reference.

AWS managed policy: AmazonS3ExpressReadOnlyAccess

You can attach the `AmazonS3ExpressReadOnlyAccess` policy to your IAM identities. This policy grants permissions that allow `ReadOnly` access to Amazon S3 Express One Zone directory buckets.

> **Note**
>
> The `CreateSession` action supports the `SessionMode` condition key which can be set to `ReadOnly` or `ReadWrite`. This policy uses `SessionMode` for a `ReadOnly` session.

To view the permissions for this policy, see AmazonS3ExpressReadOnlyAccess in the AWS Managed Policy Reference.

Amazon S3 Express One Zone updates to AWS managed policies

View details about updates to AWS managed policies for Amazon S3 Express One Zone since this service began tracking these changes.

Change	Description	Date
Amazon S3 Express One Zone added `AmazonS3ExpressFullAccess` .	Amazon S3 Express One Zone added a new AWS managed policy called `AmazonS3ExpressFullAccess` . This policy grants permissions that allow full access to Amazon S3 Express One Zone directory buckets and operations.	April 03, 2026
Amazon S3 Express One Zone added `AmazonS3ExpressReadOnlyAccess` .	Amazon S3 Express One Zone added a new AWS managed policy called `AmazonS3ExpressReadOnlyAccess` . This policy grants	April 03, 2026

Change	Description	Date
	permissions that allow read-only access to Amazon S3 Express One Zone directory buckets.	
Amazon S3 Express One Zone started tracking changes.	Amazon S3 Express One Zone started tracking changes for its AWS managed policies.	April 03, 2026

Authorizing Zonal endpoint API operations with `CreateSession`

To use Zonal endpoint API operations (object-level, or data plane operations), except for `CopyObject` and `HeadBucket`, you use the `CreateSession` API operation to create and manage sessions that are optimized for low-latency authorization of data requests. To retrieve and use a session token, you must allow the `s3express:CreateSession` action for your directory bucket in an identity-based policy or a bucket policy. For more information, see Authorizing Regional endpoint API operations with IAM. If you're accessing S3 Express One Zone in the Amazon S3 console, through the AWS Command Line Interface (AWS CLI), or by using the AWS SDKs, S3 Express One Zone creates a session on your behalf. However, you can't modify the `SessionMode` parameter when using the AWS CLI or AWS SDKs.

If you use the Amazon S3 REST API, you can then use the `CreateSession` API operation to obtain temporary security credentials that include an access key ID, a secret access key, a session token, and an expiration time. The temporary credentials provide the same permissions as long-term security credentials, such as IAM user credentials, but temporary security credentials must include a session token.

Session Mode

Session mode defines the scope of the session. If the session mode is not specified in the CreateSession API request, the CreateSession action will attempt to create the session with the maximum allowable privilege, attempting `ReadWrite` first, then falling back to `ReadOnly` only if `ReadWrite` is not permitted by the policies. In your bucket policy, you can specify the `s3express:SessionMode` condition key to explicitly control who can create a `ReadWrite` or `ReadOnly` session. For more information about `ReadWrite` or `ReadOnly` sessions, see the `x-amz-`

`create-session-mode` parameter for CreateSession in the *Amazon S3 API Reference*. For more information about the bucket policy to create, see Example bucket policies for directory buckets.

Session Token

When you make a call by using temporary security credentials, the call must include a session token. The session token is returned along with the temporary credentials. A session token is scoped to your directory bucket and is used to verify that the security credentials are valid and haven't expired. To protect your sessions, temporary security credentials expire after 5 minutes.

`CopyObject` and `HeadBucket`

Temporary security credentials are scoped to a specific directory bucket and are automatically enabled for all Zonal (object-level) operation API calls to a given directory bucket. Unlike other Zonal endpoint API operations, `CopyObject` and `HeadBucket` don't use `CreateSession` authentication. All `CopyObject` and `HeadBucket` requests must be authenticated and signed by using IAM credentials. However, `CopyObject` and `HeadBucket` are still authorized by `s3express:CreateSession`, like other Zonal endpoint API operations.

For more information, see CreateSession in the *Amazon Simple Storage Service API Reference*.

Security best practices for directory buckets

There are a number of security features to consider when working with directory buckets. The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful recommendations rather than prescriptions.

Default Block Public Access and Object Ownership settings

Directory buckets support S3 Block Public Access and S3 Object Ownership. These S3 features are used to audit and manage access to your buckets and objects.

By default, all Block Public Access settings for directory buckets are enabled. In addition, Object Ownership is set to bucket owner enforced, which means that access control lists (ACLs) are disabled. These settings can't be modified. For more information about these features, see the section called “Blocking public access” and the section called “Controlling object ownership”.

> **Note**
>
> You can't grant access to objects stored in directory buckets. You can grant access only to your directory buckets. The authorization model for S3 Express One Zone is different than the authorization model for Amazon S3. For more information, see Authorizing Zonal endpoint API operations with `CreateSession`.

Authentication and authorization

The authentication and authorization mechanisms for directory buckets differ, depending on whether you are making requests to Zonal endpoint API operations or Regional endpoint API operations. Zonal API operations are object-level (data plane) operations. Regional API operations are bucket-level (control plane) operations.

You authenticate and authorize requests to Zonal endpoint API operations through a new session-based mechanism that is optimized to provide the lowest latency. With session-based authentication, the AWS SDKs use the `CreateSession` API operation to request temporary credentials that provide low-latency access to your directory bucket. These temporary credentials are scoped to a specific directory bucket and expire after 5 minutes. You can use these temporary credentials to sign Zonal (object level) API calls. For more information, see Authorizing Zonal endpoint API operations with `CreateSession`.

Signing requests with credentials for directory bucket management

You use your credentials to sign Zonal endpoint (object level) API requests with AWS Signature Version 4, with `s3express` as the service name. When you sign your requests, use the secret key that's returned from `CreateSession` and also provide the session token with the `x-amzn-s3session-token header`. For more information, see CreateSession.

The supported AWS SDKs manage credentials and signing on your behalf. We recommend using the AWS SDKs to refresh credentials and sign requests for you.

Signing requests with IAM credentials

All Regional (bucket-level) API calls must be authenticated and signed by AWS Identity and Access Management (IAM) credentials instead of temporary session credentials. IAM credentials consist of the access key ID and secret access key for the IAM identities. All `CopyObject` and `HeadBucket` requests must also be authenticated and signed by using IAM credentials.

To achieve the lowest latency for your Zonal (object-level) operation calls, we recommend using credentials obtained from calling `CreateSession` to sign your requests, except for requests to `CopyObject` and `HeadBucket`.

Use AWS CloudTrail

AWS CloudTrail provides a record of the actions taken by a user, a role, or an AWS service in Amazon S3. You can use information collected by CloudTrail to determine the following:

- The request that was made to Amazon S3
- The IP address from which the request was made
- Who made the request
- When the request was made
- Additional details about the request

When you set up your AWS account, CloudTrail management events are enabled by default. The following Regional endpoint API operations (bucket-level, or control plane, API operations) are logged to CloudTrail.

- CreateBucket
- DeleteBucket
- DeleteBucketPolicy
- PutBucketPolicy
- GetBucketPolicy
- ListDirectoryBuckets
- ListMultipartUploads
- PutBucketEncryption
- GetBucketEncryption
- DeleteBucketEncryption

> **Note**
>
> `ListMultipartUploads` is a Zonal endpoint API operation. However, it is logged to CloudTrail as a management event. For more information, see ListMultipartUploads in the *Amazon Simple Storage Service API Reference*.

By default, CloudTrail trails don't log data events, but you can configure trails to log data events for directory buckets that you specify, or to log data events for all the directory buckets in your AWS account. The following Zonal endpoint API operations (object-level, or data plane, API operations) are logged to CloudTrail.

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateSession
- CopyObject
- CreateMultipartUpload
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- ListParts
- PutObject
- UploadPart
- UploadPartCopy

For more information on using AWS CloudTrail with directory buckets , see Logging with AWS CloudTrail for directory buckets.

Implement monitoring by using AWS monitoring tools

Monitoring is an important part of maintaining the reliability, security, availability, and performance of Amazon S3 and your AWS solutions. AWS provides several tools and services to help you monitor Amazon S3 and your other AWS services. For example, you can monitor Amazon CloudWatch metrics for Amazon S3, particularly the `BucketSizeBytes` and `NumberOfObjects` storage metrics.

Objects stored in the directory buckets won't be reflected in the `BucketSizeBytes` and `NumberOfObjects` storage metrics for Amazon S3. However, the `BucketSizeBytes` and `NumberOfObjects` storage metrics are supported for directory buckets. To see the metrics of your choice, you can differentiate between the Amazon S3 storage classes by specifying a `StorageType` dimension. For more information, see Monitoring metrics with Amazon CloudWatch.

For more information, see Monitoring metrics with Amazon CloudWatch and Logging and monitoring in Amazon S3.

Managing access to shared datasets in directory buckets with access points

Amazon S3 Access Points simplify managing data access at scale for shared datasets in Amazon S3. Access points are unique hostnames you create to enforce distinct permissions and network controls for all requests made through an access point. You can create hundreds of access points per bucket, each with a distinct name and permissions customized for each application. Each access point works in conjunction with the bucket policy that is attached to the underlying bucket.

In directory buckets, an access point name consists of a base name you provide, followed by the Zone ID (AWS Availability Zone or Local Zone) of your directory bucket location, and then `--xa-s3`. For example, *`accesspointname`*`--`*`zoneID`*`--xa-s3`. After you create an access point, you can't change the name or the Zone ID.

With access points for directory buckets, you can use the access point scope to restrict access to specific prefixes or API operations. You can specify any amount of prefixes, but the total length of characters of all prefixes must be less than 256 bytes.

You can configure any access point to accept requests only from a virtual private cloud (VPC). This restricts Amazon S3 data access to a private network.

In this section, the topics explain how to use access points for directory buckets. For information about directory buckets, see Working with directory buckets.

Topics

- Access points for directory buckets naming rules, restrictions, and limitations
- Referencing access points for directory buckets
- Object operations for access points for directory buckets
- Configuring IAM policies for using access points for directory buckets
- Monitoring and logging access points for directory buckets
- Creating access points for directory buckets
- Managing your access points for directory buckets
- Using tags with S3 Access Points for directory buckets

Access points for directory buckets naming rules, restrictions, and limitations

Access points simplify managing data access at scale for shared datasets in Amazon S3. The following topics provide information about access point naming rules and restrictions and limitations.

Topics

- Naming rules for access points for directory buckets
- Restrictions and limitations for access points for directory buckets

Naming rules for access points for directory buckets

An access point must be created in the same zone that the bucket is in. An access point name must be unique within the zone.

Access point names must be DNS-compliant and must meet the following conditions:

- Must begin with a number or lowercase letter
- The base name you provide must be between 3 and 50 characters long
- Can't begin or end with a hyphen (-)

- Can't contain underscores (_), uppercase letters, spaces, or periods (.)
- Must end with the suffix *zoneid*`--xa--s3`.

Restrictions and limitations for access points for directory buckets

Access points for directory buckets have the following restrictions and limitations:

- Each access point is associated to one directory bucket. After you create an access point, you can't associate it to a different bucket. However, you can delete an access point, and then create a new one with the same name and associate it to a different bucket.
- After you create an access point, you can't change its virtual private cloud (VPC) configuration.
- Access point policies are limited to 20 KB in size.
- Access point scope prefixes are limited to 256 bytes in total size.
- You can create a maximum of 10,000 access points per AWS account per AWS Region. If you need more than 10,000 access points for a single account in a single Region, you can request a service quota increase. For more information about service quotas and requesting an increase, see AWS service quotas in the *AWS General Reference*.
- You can only use access points to perform operations on objects. You can't use access points to perform Amazon S3 bucket operations, such as modifying or deleting buckets. For a complete list of supported operations, see Object operations for access points for directory buckets.
- You can refer to access points by name, access point alias, or virtual-hosted-style URI. You cannot address access points by ARN. For more information, see Referencing access points for directory buckets.
- API operations that control access point functionality (for example, `PutAccessPointPolicy` and `GetAccessPointPolicy`) must specify the AWS account that owns the access point.
- You must use AWS Signature Version 4 when making requests to an access point by using the REST API. For more information about authenticating requests, see Authenticating Requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.
- Access points only support requests over HTTPS. Amazon S3 will automatically respond with an HTTP redirect for any requests made through HTTP, to upgrade the request to HTTPS.
- Access points don't support anonymous access.
- If you create an access point to a bucket that's owned by another account (a cross-account access point), the cross-account access point doesn't grant you access to data until the bucket owner grants you permission to access the bucket. The bucket owner always retains ultimate control

over access to the data and must update the bucket policy to authorize requests from the cross-account access point. To view a bucket policy example, see Configuring IAM policies for using access points for directory buckets.

Referencing access points for directory buckets

After you create an access point, you can use it as an endpoint to preform object operations. For access points for directory buckets, the access point alias is the same as the access point name. You can use the access point name instead of a bucket name for all data operations. For a list of these supported operations, see Object operations for access points for directory buckets.

Referring to access points by virtual-hosted-style URIs

Access points only support virtual-host-style addressing. Access points use the same format as directory bucket endpoints. For more information, see Regional and Zonal endpoints for directory buckets.

S3 access points don't support access through HTTP. Access points support only secure access through HTTPS.

Object operations for access points for directory buckets

You can use access points to access an object using the following S3 data operations.

- AbortMultipartUpload
- CompleteMultipartUpload
- CopyObject
- CreateMultipartUpload
- CreateSession
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject

- ListMultipartUploads
- ListObjectsV2
- ListParts
- PutObject
- UploadPart
- UploadPartCopy

Configuring IAM policies for using access points for directory buckets

Access points support AWS Identity and Access Management (IAM) resource policies that allow you to control the use of the access point by resource, user, or other conditions. For an application or user to access objects through an access point, both the access point and the underlying bucket policy must permit the request.

> **Important**
>
> Adding an access point to a directory bucket doesn't change the bucket's behavior when the bucket is accessed directly through the bucket's name. All existing operations against the bucket will continue to work as before. Restrictions that you include in an access point policy or access point scope apply only to requests made through that access point.

When using IAM resource policies, make sure to resolve security warnings, errors, general warnings, and suggestions from AWS Identity and Access Management Access Analyzer before you save your policy. IAM Access Analyzer runs policy checks to validate your policy against IAM policy grammar and best practices. These checks generate findings and provide recommendations to help you author policies that are functional and conform to security best practices.

To learn more about validating policies by using IAM Access Analyzer, see IAM Access Analyzer policy validation in the *IAM User Guide*. To view a list of the warnings, errors, and suggestions that are returned by IAM Access Analyzer, see IAM Access Analyzer policy check reference.

Access points for directory buckets policy examples

The following access point policies demonstrate how to control requests to a directory bucket. Access point policies require bucket ARNs or access point ARNs. Access point aliases are not supported in policies. Following is an example of an access point ARN:

```
arn:aws:s3express:region:account-id:accesspoint/myaccesspoint--zoneID--xa-s3
```

You can view the access point ARN in the details of an access point. For more information, see View details for your access points for directory buckets.

Note

Permissions granted in an access point policy are effective only if the underlying bucket also allows the same access. You can accomplish this in two ways:

1. **(Recommended)** Delegate access control from the bucket to the access point, as described in Delegating access control to access points.
2. Add the same permissions contained in the access point policy to the underlying bucket's policy.

Example 1 – Service control policy to limit access points to VPC network origins

The following service control policy requires all new access points are to be created with a virtual private cloud (VPC) network origin. With this policy in place, users in your organization can't create any access point that is accessible from the internet.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Deny",
        "Action": "s3express:CreateAccessPoint",
        "Resource": "*",
        "Condition": {
            "StringNotEquals": {
                "s3express:AccessPointNetworkOrigin": "VPC"
            }
        }
    }
  ]
```

```
}
```

Example 2 – Access point policy to limit bucket access to access points with VPC network origin

The following access point policy limits all access to the bucket *amzn-s3-demo-bucket--zoneID--x-s3* to an access point with a VPC networking origin.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DenyCreateSessionFromNonVPC",
            "Principal": "*",
            "Action": "s3express:CreateSession",
            "Effect": "Deny",
            "Resource": "arn:aws:s3express:us-east-1:111122223333:bucket/amzn-s3-demo-bucket--usw2-az1--x-s3"
        }
    ]
}
```

Condition keys

Access points for directory buckets have condition keys that you can use in IAM policies to control access to your resources. The following condition keys represent only part of an IAM policy. For full policy examples, see Access points for directory buckets policy examples, Delegating access control to access points, and Granting permissions for cross-account access points.

`s3express:DataAccessPointArn`

This example shows how to filter access by the Amazon resource name (ARN) of an access point and matches all access points for AWS account *111122223333* in Region *region*:

```
"Condition" : {
    "StringLike": {
        "s3express:DataAccessPointArn":
 "arn:aws:s3express:region:111122223333:accesspoint/*"
```

```
    }
}
```

s3express:DataAccessPointAccount

This example shows a string operator that you can use to match on the account ID of the owner of an access point. The following example matches all access points that are owned by the AWS account *111122223333*.

```
"Condition" : {
    "StringEquals": {
        "s3express:DataAccessPointAccount": "111122223333"
    }
}
```

s3express:AccessPointNetworkOrigin

This example shows a string operator that you can use to match on the network origin, either `Internet` or VPC. The following example matches only access points with a VPC origin.

```
"Condition" : {
    "StringEquals": {
        "s3express:AccessPointNetworkOrigin": "VPC"
    }
}
```

s3express:Permissions

You can use `s3express:Permissions` to restrict access to specific API operations in access point scope. The following API operations are supported:

- `PutObject`
- `GetObject`
- `DeleteObject`
- `ListBucket` (required for `ListObjectsV2`)
- `GetObjectAttributes`
- `AbortMultipartUpload`
- `ListBucketMultipartUploads`
- `ListMultipartUploadParts`

> **Note**
>
> When using multi-value condition keys, we recommend you use `ForAllValues` with `Allow` statements and `ForAnyValue` with `Deny` statements. For more information, see Multivalued context keys in the IAM User Guide.

For more information about using condition keys with Amazon S3, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the required permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Delegating access control to access points

You can delegate access control from the bucket policy to the access point policy. The following example bucket policy allows full access to all access points that are owned by the bucket owner's account. After applying the policy, all access to this bucket is controlled by access point policies. We recommend configuring your buckets this way for all use cases that don't require direct access to the bucket.

Example bucket policy that delegates access control to access points

```
{
    "Version": "2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "Bucket ARN",
        "Condition": {
            "StringEquals" : { "s3express:DataAccessPointAccount" : "Bucket owner's
 account ID" }
        }
    }]
}
```

Granting permissions for cross-account access points

To create an access point to a bucket that's owned by another account, you must first create the access point by specifying the bucket name and account owner ID. Then, the bucket owner must update the bucket policy to authorize requests from the access point. Creating an access point is similar to creating a DNS CNAME in that the access point doesn't provide access to the bucket contents. All bucket access is controlled by the bucket policy. The following example bucket policy allows `GET` and `LIST` requests on the bucket from an access point that's owned by a trusted AWS account.

Replace *`Bucket ARN`* with the ARN of the bucket.

Example of bucket policy delegating permissions to another AWS account

JSON

```
{
    "Version":"2012-10-17",
    "Statement" : [
    {
        "Sid": "AllowCreateSessionForDirectoryBucket",
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "s3express:CreateSession",
        "Resource" : [ "arn:aws:s3express:us-west-2:111122223333:bucket/amzn-s3-demo-bucket--usw2-az1--x-s3" ],
        "Condition": {
            "ForAllValues:StringEquals": {
                "s3express:Permissions": [
                    "GetObject",
                    "ListBucket"
                ]
            }
        }
    }]
}
```

Monitoring and logging access points for directory buckets

You can log requests made through access points and requests made to the APIs that manage access points, such as `CreateAccessPoint` and `GetAccessPointPolicy`, by using AWS CloudTrail. CloudTrail log entries for requests made through access points include the access point ARN (which includes the access point name) in the `resources` section of the log.

For example, suppose you have the following configuration:

- A bucket named *amzn-s3-demo-bucket*`--`*zone-id*`--x-s3` in Region `region` that contains an object named `my-image.jpg`.
- An access point named `my-bucket-ap--zoneID--xa-s3` that is associated with *amzn-s3-demo-bucket*`--`*zone-id*`--x-s3`
- An AWS account ID of `123456789012`

The following example shows the `resources` section of a CloudTrail log entry for the preceding configuration:

```
"resources": [
        {"type": "AWS::S3Express::Object",

            "ARN": "arn:aws:s3express-region:123456789012:bucket/amzn-s3-demo-
bucket--zone-id--x-s3/my-image.jpg"
        },
        {"accountId": "c",
            "type": "AWS::S3Express::DirectoryBucket",
            "ARN": "arn:aws::s3express:region:123456789012:bucket/amzn-s3-demo-
bucket--zone-id--x-s3"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::AccessPoint",
            "ARN": "arn:aws:s3express:region:123456789012:accesspoint/my-bucket-ap--
zoneID--xa-s3"
        }
    ]
```

For more information about AWS CloudTrail, see What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

Creating access points for directory buckets

Like Directory buckets, access points can be created in Availability Zones or in Dedicated Local Zones. The access point must be created in the same zone as the directory bucket associated with it.

An access point is associated with exactly one Amazon S3 directory bucket. If you want to use a directory bucket in your AWS account, you must first create a directory bucket. For more information about creating directory buckets, see Creating directory buckets in an Availability Zone or Creating a directory bucket in a Local Zone.

You can also create a cross-account access point that's associated with a bucket in another AWS account, as long as you know the bucket name and the bucket owner's account ID. However, creating cross-account access points doesn't grant you access to data in the bucket until you are granted permissions from the bucket owner. The bucket owner must grant the access point owner's account (your account) access to the bucket through the bucket policy. For more information, see Granting permissions for cross-account access points.

You can create an access point for any directory bucket with the AWS Management Console, AWS CLI, REST API, or AWS SDKs. Each access point is associated with a single directory bucket, and you can create hundreds of access points per bucket. When creating an access point, you choose the name of the access point and the directory bucket to associate it with. The access point name consists of a base name that you provide and suffix that includes the Zone ID of your bucket location, followed by `--xa-s3`. For example, *`myaccesspoint-zoneID`*`--xa-s3`. you can also restrict access to the access point through a Virtual Private Cloud (VPC). Then, you can immediately begin reading and writing data through your access point by using its name, just like you use a directory bucket name.

You can use the access point scope to restrict access to the directory bucket through the access point to specific prefixes or API operations. If you don't add a scope to the access point, all prefixes in the directory bucket and all API operations can be performed on objects in the bucket when accessed through the access point. After you create the access point, you can add, modify, or delete scope using the AWS CLI, AWS SDKs, or REST API. For more information, see Manage the scope of your access points for directory buckets.

After you create the access point, you can configure your access point IAM resource policy. For more information, see Viewing, editing or deleting access point policies.

Using the S3 console

> **Note**
>
> You can also create an access point for a directory bucket from the directory bucket screen. When you do this, the directory bucket name is provided and you don't need to choose a bucket when creating the access point. For more information, see Listing directory buckets.

To create an access point for directory buckets

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create an access point. The access point must be created in the same Region as the associated bucket.
3. In the left navigation pane, choose **Access points for directory buckets**.
4. On the **Access Points** page, choose **Create access point**.
5. You can create an access point for a directory bucket in your account or in another account. To create an access point for a directory bucket in another account:

   > **Note**
   >
   > If you're using a bucket in a different AWS account, the bucket owner must update the bucket policy to authorize requests from the access point. For an example bucket policy, see Granting permissions for cross-account access points.

   a. In the **Directory bucket** field, choose **Specify a bucket in another account**.

   b. In the **Bucket owner account ID** field, enter the AWS account ID that owns the bucket.

   c. In the **Bucket name** field, enter the name of the bucket, including the base name and the zone ID. For example, ***bucket-base-name*--*zone-id*--x-s3**.

6. To create an access point for a directory bucket in your account:

   a. In the **Directory bucket** field, choose **Choose a bucket in this account**.

b. In the **Bucket name** field, enter the name of the bucket, including the base name and the zone ID. For example, ***bucket-base-name*--*zone-id*--x-s3**. To choose the bucket from a list, choose **Browse S3** and choose the directory bucket.

7. In **Access point name**, in the **Base name** field, enter the base name for the access point. The zone ID and full access point name appear. For more information about naming access points, see Naming rules for access points for directory buckets.
8. In **Network origin**, choose either **virtual private cloud (VPC)** or **Internet**. If you choose **virtual private cloud (VPC)**, in the **VPC ID** field, enter the ID of the VPC that you want to use with the access point.
9. (Optional) In **Access point scope**, to apply a scope to this access point, choose **Limit the scope of this access point using prefixes or permissions**.

   a. To limit access to prefixes in the directory bucket, in **Prefixes**, enter one or more prefixes. To add another prefix, choose **Add prefix**. To remove a prefix, choose **Remove**.

   > **Note**
   >
   > An access point scope has a character limit of 512 total characters for all prefixes. You can see the quantity of characters remaining below **Add prefix**.

   b. In **Permissions**, choose one or more API operations that the access point will allow. To remove a data operation, choose the **X** next to the data operation name.

10. To not apply a scope to the access point and allow access to all prefixes in the directory bucket and all API operations through the access point, in **Access point scope**, choose **Apply access to the entire bucket**.
11. Choose **Create access point for directory bucket**. The access point name and other information about it appear in the **Access points for directory buckets** list.

Using the AWS CLI

The following example command creates an access point named *example-ap* for the bucket *amzn-s3-demo-bucket--zone-id--x-s3* in the account *111122223333*.

```
aws s3control create-access-point --name example-ap--zoneID--xa-s3 --account-
id 111122223333 --bucket amzn-s3-demo-bucket--zone-id--x-s3
```

To restrict access to the access point through a VPC, include the `--vpc` parameter and the VPC ID.

```
aws s3control create-access-point --name example-ap--zoneID--xa-s3 --account-
id 111122223333 --bucket amzn-s3-demo-bucket--zone-id--x-s3 --vpc vpc-id
```

When you create an access point for a cross-account bucket, include the `--bucket-account-id` parameter. The following example command creates an access point in the AWS account *111122223333*, for the bucket *amzn-s3-demo-bucket--zone-id--x-s3*, owned by the AWS account *444455556666*.

```
aws s3control create-access-point --name example-ap--zoneID--xa-s3 --account-
id 111122223333 --bucket amzn-s3-demo-bucket--zone-id--x-s3 --bucket-account-
id 444455556666
```

For more information and examples, see create-access-point in the AWS CLI Command Reference.

Using the REST API

The following example command creates an access point named *example-ap* for the bucket *amzn-s3-demo-bucket--zone-id--x-s3* in the account *111122223333* and access restricted through the VPC *vpc-id* (optional).

```
PUT /v20180820/accesspoint/example-ap--zoneID--xa-s3 HTTP/1.1
Host: s3express-control.region.amazonaws.com
x-amz-account-id: 111122223333
<?xml version="1.0" encoding="UTF-8"?>
<CreateAccessPointRequest>
   <Bucket>amzn-s3-demo-bucket--zone-id--x-s3s</Bucket>
   <BucketAccountId>111122223333</BucketAccountId>
   <VpcConfiguration>
       <VpcId>vpc-id</VpcId>
   </VpcConfiguration>
</CreateAccessPointRequest>
```

Response:

```
HTTP/1.1 200
<?xml version="1.0" encoding="UTF-8"?>
<CreateAccessPointResult>
   <AccessPointArn>
```

```
        "arn:aws:s3express:region:111122223333:accesspoint/example-ap--zoneID--xa-s3"
    </AccessPointArn>
    <Alias>example-ap--zoneID--xa-s3</Alias>
</CreateAccessPointResult>
```

Using the AWS SDKs

You can use the AWS SDKs to create an access point. For more information, see list of supported SDKs in the Amazon Simple Storage Service API Reference.

Managing your access points for directory buckets

This section explains how to manage your access points for directory buckets using the AWS Command Line Interface, Amazon S3 REST API, or AWS SDK.

Topics

- List your access points for directory buckets
- View details for your access points for directory buckets
- Viewing, editing or deleting access point policies
- Manage the scope of your access points for directory buckets
- Delete your access point for directory buckets

List your access points for directory buckets

This section explains how to list access points for a directory bucket using the AWS Management Console, AWS Command Line Interface (AWS CLI), REST API, or AWS SDKs.

Using the S3 console

To list access points in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access points for directory buckets**.

4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage.

Using the AWS CLI

The following `list-access-points-for-directory-buckets` example command shows how you can use the AWS CLI to list the access points owned by an AWS account and associated with a directory bucket.

The following command lists access points for AWS account *111122223333* that are attached to bucket *amzn-s3-demo-bucket--zone-id--x-s3*.

```
aws s3control list-access-points-for-directory-buckets --account-id 111122223333 --
directory-bucket amzn-s3-demo-bucket--zone-id--x-s3
```

For more information and examples, see list-access-points-for-directory-buckets in the AWS CLI Command Reference.

Using the REST API

The following example shows how you can use the REST API to list your access points.

```
GET /v20180820/directoryaccesspoint?directoryBucket=amzn-s3-demo-bucket--zone-id--x-s3
&maxResults=maxResults HTTP/1.1
Host: s3express-control.region.amazonaws.com
x-amz-account-id: 111122223333
```

Example of `ListAccessPointsForDirectoryBuckets` response

```
HTTP/1.1 200
<?xml version="1.0" encoding="UTF-8"?>
<ListDirectoryAccessPointsResult>
    <AccessPointList>
        <AccessPoint>
            <AccessPointArn>arn:aws:s3express:region:111122223333:accesspoint/example-
access-point--zoneID--xa-s3</AccessPointArn>
```

```
            <Alias>example-access-point--zoneID--xa-s3</Alias>
            <Bucket>amzn-s3-demo-bucket--zone-id--x-s3</Bucket>
            <BucketAccountId>111122223333</BucketAccountId>
            <Name>example-access-point--zoneID--xa-s3</Name>
            <NetworkOrigin>VPC</NetworkOrigin>
            <VpcConfiguration>
                <VpcId>VPC-1</VpcId>
            </VpcConfiguration>
        </AccessPoint>
    </AccessPointList>
</ListDirectoryAccessPointsResult>
```

Using the AWS SDKs

You can use the AWS SDKs to list your access points. For more information, see list of supported SDKs in the Amazon Simple Storage Service API Reference.

View details for your access points for directory buckets

This section explains how to view details for your access point for directory buckets using the AWS Management Console, AWS CLI, AWS SDKs, or REST API.

Using the S3 console

View details of an access point for directory buckets to see the following information about the access point and the associated directory bucket:

- Properties:
  - Directory bucket name
  - Directory bucket owner account ID
  - AWS Region
  - Directory bucket location type
  - Directory bucket location name
  - Creation date of access point
  - Network origin
  - VPC ID
  - S3 URI
  - Access point ARN

- Access point alias
- Permissions:
  - IAM external access analyzer findings
  - Access point scope
  - Access point policy

To view details for your access point in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access points for directory buckets**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage.
6. Choose the **Properties** tab or the **Permissions** tab.

Using the AWS CLI

The following `get-access-point` example command shows how you can use the AWS CLI to view details for your access point.

The following command lists details for the access point *my-access-point--zoneID--xa-s3* for AWS account *111122223333*.

```
aws s3control get-access-point --name my-access-point--zoneID--xa-s3 --account-
id 111122223333
```

Example of output of `get-access-point` command

```
{
    "Name": "example-access-point--zoneID--xa-s3",
```

```
    "Bucket": "amzn-s3-demo-bucket--zone-id--x-s3",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2025-04-23T18:26:22.146000+00:00",
    "Alias": "example-access-point--zoneID--xa-s3",
    "AccessPointArn": "arn:aws:s3express:region:111122223333:accesspoint/example-
access-point--zoneID--xa-s3",
    "BucketAccountId": "296805379465"
}
```

For more information and examples, see get-access-point in the *AWS CLI Command Reference*.

Using the REST API

You can use the REST API to view details for your access point. For more information, see GetAccessPoint in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

You can use the AWS SDKs to view details of your access points. For more information, see list of supported SDKs in the Amazon Simple Storage Service API Reference.

Viewing, editing or deleting access point policies

You can use an AWS Identity and Access Management (IAM) access point policy to control the principal and resource that can access the access point. The access point scope manages the prefixes and API permissions for the access point. You can create, edit, and delete an access point policy using the AWS Command Line Interface, REST API, or AWS SDKs. For more information about access point scope, see Manage the scope of your access points for directory buckets.

> **Note**
>
> Since directory buckets use session-based authorization, your policy must always include the `s3express:CreateSession` action.

Using the S3 console

To view, edit, or delete an access point policy

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access points for directory buckets**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage.
6. Choose the **Permissions** tab.
7. To create or edit the access point policy, in **Access point policy**, choose **Edit**. Edit the policy. Choose **Save**.
8. To delete the access point policy, in **Access point policy**, choose **Delete**. In the **Delete access point policy** window, type **`confirm`** and choose **Delete**.

Using the AWS CLI

You can use the `get-acccess-point-policy`, `put-access-point-policy`, and `delete-access-point-policy` commands to view, edit, or delete an access point policy. For more information, see get-access-point-policy, put-access-point-policy, or delete-access-point-policy in the AWS CLI Command Reference.

Using the REST API

You can use the REST API `GetAccessPointPolicy`, `DeleteAccessPointPolicy`, and `PutAccessPointPolicy` operations to view, delete, or edit an access point policy. For more information, see PutAccessPointPolicy, GetAccessPointPolicy, or DeleteAccessPointPolicy in the Amazon Simple Storage Service API Reference.

Using the AWS SDKs

You can use the AWS SDKs to view, delete, or edit an access point policy. For more information, see the list of supported SDKs for GetAccessControlPolicy, DeleteAccessControlPolicy, and PutAccessControlPolicy in the Amazon Simple Storage Service API Reference.

Manage the scope of your access points for directory buckets

This section explains how to view and modify the scope of your access points for directory buckets using the AWS Command Line Interface, REST API, or AWS SDKs. You can use the access point scope to restrict access to specific prefixes or API operations.

Topics

- View the scope of your access points for directory buckets
- Modify the scope of your access point for directory buckets
- Delete the scope of your access points for directory buckets

View the scope of your access points for directory buckets

You can use the AWS Management Console, AWS Command Line Interface, REST API, or AWS SDKs to view the scope of your access point for directory buckets.

Using the S3 console

To view the scope of your access point for directory buckets

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access points for directory buckets**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage.
6. Choose the **Permissions** tab.
7. In the **Access point scope**, you can see the prefixes and permissions applied to the access point.

Using the AWS CLI

The following `get-access-point-scope` example command shows how you can use the AWS CLI to view the scope of your access point.

The following command shows the scope of the access point *my-access-point*--*zoneID*--xa-s3 for AWS account *111122223333*.

```
aws s3control get-access-point-scope --name my-access-point--zoneID--xa-s3 --account-
id 111122223333
```

For more information and examples, see get-access-point-scope in the AWS CLI Command Reference.

Example result of `get-access-point-scope`

```
{
    "Scope": {
        "Permissions": [
            "ListBucket",
            "PutObject"
        ]
  "Prefixes": [
            "Prefix": "MyPrefix1*",
            "Prefix": "MyObjectName.csv"
        ]
    }
}
```

Using the REST API

The following `GetAccessPointScope` example request shows how you can use the REST API to view the scope of your access point.

The following request shows the scope of the access point *my-access-point*--*region*-*zoneID*--xa-s3 for AWS account *111122223333*.

```
GET /v20180820/accesspoint/my-access-point--zoneID--xa-s3/scope HTTP/1.1
Host: s3express-control.region.amazonaws.com
x-amz-account-id: 111122223333
```

Example result of `GetAccessPointScope`

```
HTTP/1.1 200
<?xml version="1.0" encoding="UTF-8"?>
<GetAccessPointScopeResult>
    <Scope>
        <Prefixes>
            <Prefix>MyPrefix1*</Prefix>
            <Prefix>MyObjectName.csv</Prefix>
        </Prefixes>
        <Permissions>
            <Permission>ListBucket</Permission>
            <Permission>PutObject</Permission>
        </Permissions>
        <Scope>
</GetAccessPointScopeResult>
```

Using the AWS SDKs

You can use the AWS SDKs to view the scope of your access point. For more information, see list of supported SDKs in the Amazon Simple Storage Service API Reference.

Modify the scope of your access point for directory buckets

You can use the AWS Management Console, AWS Command Line Interface, REST API, or AWS SDKs to modify the scope of your access points for directory buckets. Access point scope is used to restrict access to specific prefixes, API operations, or a combination of both.

You can include one or more of the following API operations as permissions:

- `PutObject`
- `GetObject`
- `DeleteObject`
- `ListBucket` (required for `ListObjectsV2`)
- `GetObjectAttributes`
- `AbortMultipartUploads`
- `ListBucketMultipartUploads`
- `ListMultipartUploadParts`

> **Note**
>
> You can specify any amount of prefixes, but the total length of characters of all prefixes must be less than 256 bytes in size.

Using the S3 console

To modify access point scope

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access points for directory buckets**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage.
6. Choose the **Permissions** tab.
7. In the **Access point scope** section, choose **Edit**.
8. To add or remove prefixes:
    a. To add a prefix, choose **Add prefix**. In the **Prefix** field, enter a prefix of the directory bucket. Repeat to add more prefixes.
    b. To remove a prefix, choose **Remove**.
9. To add or remove permissions:
    a. To add a permission, in the **Choose data operations** field, choose the permission.
    b. To remove a permission, choose the **X** next to the permission.
10. Choose **Save changes**.

Using the AWS CLI

The following `put-access-point-scope` example command shows how you can use the AWS CLI to modify the scope of your access point.

The following command modifies the access point scope of *my-access-point*--*zoneID*--xa-s3 for AWS account *111122223333*.

> **Note**
>
> You can use wildcards in prefixes by using the asterisk (*) character. If you want to use the asterisk character as a literal, add a backslash character (\) before it to escape it.
> All prefixes have an implicit '*' ending, meaning all paths withing the prefix will be included.
> When you modify the scope of an access point with the AWS CLI, you replace the existing scope.

```
aws s3control put-access-point-scope --name my-access-point--zoneID--xa-s3 --account-
id 111122223333 --scope Prefixes=string,Permissions=string
```

For more information and examples, see put-access-point-scope in the AWS CLI Command Reference.

Using the REST API

The following PutAccessPointScope example request shows how you can use the REST API to modify the scope of your access point.

The following request modifies the access point scope of *my-access-point*--*zoneID*--xa-s3 for AWS account *111122223333*.

> **Note**
>
> You can use wildcards in prefixes by using the asterisk (*) character. If you want to use the asterisk character as a literal, add a backslash character (\) before it to escape it.
> All prefixes have an implicit '*' ending, meaning all paths withing the prefix will be included.
> When you modify the scope of an access point with the API, you replace the existing scope.

```
PUT /v20180820/accesspoint/my-access-point--zoneID--xa-s3/scope HTTP/1.1
Host: s3express-control.region.amazonaws.com
x-amz-account-id: 111122223333
<?xml version="1.0" encoding="UTF-8"?>
```

```
<PutAccessPointScopeRequest>
        <Scope>
            <Prefixes>
                <Prefix>Jane/*</Prefix>
            </Prefixes>
            <Permissions>
                <Permission>PutObject</Permission>
                <Permission>GetObject</Permission>
            </Permissions>
            <Scope>
    </PutAccessPointScopeRequest>
```

Using the AWS SDKs

You can use the AWS CLI, AWS SDKs, or REST API to modify the scope of your access point. For more information, see list of supported SDKs in the Amazon Simple Storage Service API Reference.

Delete the scope of your access points for directory buckets

You can use the AWS Management Console, AWS Command Line Interface, REST API, or AWS SDKs to delete the scope of your access points for directory buckets.

> **Note**
>
> When you delete the scope of an access point, all prefixes and permissions are deleted.

Using the S3 console

To delete access point scope

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access points for directory buckets**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage.

6. Choose the **Permissions** tab.
7. In **Access point scope**, choose **Delete**.
8. In the **to confirm this deletion, type "confirm".** field, enter **`confirm`**.
9. Choose **Delete**.

Using the AWS CLI

The following `delete-access-point-scope` example command shows how you can use the AWS CLI to delete the scope of your access point.

The following command deletes the scope of the access point *my-access-point*--*zoneID*--xa-s3 for AWS account *111122223333*.

```
aws s3control delete-access-point-scope --name my-access-point--region-zoneID--xa-s3 --
account-id 111122223333
```

For more information and examples, see delete-access-point-scope in the AWS CLI Command Reference.

Using the REST API

The following request deletes the scope of the access point *my-access-point*--*zoneID*--xa-s3 for AWS account *111122223333*.

```
DELETE /v20180820/accesspoint/my-access-point--zoneID--xa-s3/scope HTTP/1.1
Host: s3express-control.region.amazonaws.com
x-amz-account-id: 111122223333
```

Using the AWS SDKs

You can use the AWS SDKs to delete the scope of your access point. For more information, see list of supported SDKs in the Amazon Simple Storage Service API Reference.

Delete your access point for directory buckets

This section explains how to delete your access point using the AWS Management Console, AWS Command Line Interface, REST API, or AWS SDKs.

> **Note**
>
> Before you can delete a directory bucket attached to an access point, you must delete the access point.

Using the S3 console

To delete access points for directory buckets in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access points for directory buckets**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to delete.
6. Choose **Delete**.
7. To confirm deletion, type **`confirm`** and choose **Delete**.

Using the AWS CLI

The following `delete-access-point` example command shows how you can use the AWS CLI to delete your access point.

The following command deletes the access point *`my-access-point`*--*`zoneID`*--xa-s3 for AWS account *`111122223333`*.

```
aws s3control delete-access-point --name my-access-point--zoneID--xa-s3 --account-
id 111122223333
```

For more information and examples, see delete-access-point in the *AWS CLI Command Reference*.

Using the REST API

You can use the REST API to delete your access point. For more information, see DeleteAccessPoint in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

You can use the AWS SDKs to delete your access points. For more information, see list of supported SDKs in the Amazon Simple Storage Service API Reference.

Using tags with S3 Access Points for directory buckets

An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 Access Points for directory buckets. You can tag access points when you create them or manage tags on existing access points. For general information about tags, see Tagging for cost allocation or attribute-based access control (ABAC).

> **Note**
>
> There is no additional charge for using tags on access points for directory buckets beyond the standard S3 API request rates. For more information, see Amazon S3 pricing.

Common ways to use tags with access points for directory buckets

Attribute-based access control (ABAC) allows you to scale access permissions and grant access to access points for directory buckets based on their tags. For more information about ABAC in Amazon S3, see Using tags for ABAC.

ABAC for S3 Access Points

Amazon S3 Access Points support attribute-based access control (ABAC) using tags. Use tag-based condition keys in your AWS organizations, IAM, and Access Points policies. For enterprises, ABAC in Amazon S3 supports authorization across multiple AWS accounts.

In your IAM policies, you can control access to access points for directory buckets based on the bucket's tags by using the following global condition keys:

- `aws:ResourceTag/key-name`
  - Use this key to compare the tag key-value pair that you specify in the policy with the key-value pair attached to the resource. For example, you could require that access to a resource

is allowed only if the resource has the attached tag key `Dept` with the value `Marketing`. For more information, see Controlling access to AWS resources.

- `aws:RequestTag/key-name`
  - Use this key to compare the tag key-value pair that was passed in the request with the tag pair that you specify in the policy. For example, you could check whether the request includes the tag key `Dept` and that it has the value `Accounting`. For more information, see Controlling access during AWS requests. You can use this condition key to restrict which tag key-value pairs can be passed during the `TagResource` and `CreateAccessPoint` API operations.
- `aws:TagKeys`
  - Use this key to compare the tag keys in a request with the keys that you specify in the policy. We recommend that when you use policies to control access using tags, use the `aws:TagKeys` condition key to define what tag keys are allowed. For example policies and more information, see Controlling access based on tag keys. You can create an access point for directory buckets with tags. To allow tagging during the `CreateAccessPoint` API operation, you must create a policy that includes both the `s3express:TagResource` and `s3express:CreateAccessPoint` actions. You can then use the `aws:TagKeys` condition key to enforce using specific tags in the `CreateAccessPoint` request.
- `s3express:AccessPointTag/tag-key`
  - Use this condition key to grant permissions to specific data via access points using tags. When using `aws:ResourceTag/tag-key` in an IAM policy, both the access point as well as the bucket to which the access point points to are required to have the same tag as they are both considered during authorization. If you want to control access to your data specifically via the access-point tag only, you can use `s3express:AccessPointTag/tag-key` condition key.

Example ABAC policies for access points for directory buckets

See the following example ABAC policies for access points for directory buckets.

1.1 - IAM policy to create or modify access points with specific tags

In this IAM policy, users or roles with this policy can only create access points if they tag the access points with the tag key `project` and tag value `Trinity` in the access point creation request. They can also add or modify tags on existing access points for directory buckets as long as the `TagResource` request includes the tag key-value pair `project:Trinity`.

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "CreateAccessPointWithTags",
      "Effect": "Allow",
      "Action": [
        "s3express:CreateAccessPoint",
        "s3express:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/project": [
            "Trinity"
          ]
        }
      }
    }
  ]
}
```

1.2 - Access Point policy to restrict operations on the bucket using tags

In this Access Point policy, IAM principals (users and roles) can perform operations using the `CreateSession` action on the access point only if the value of the access point's `project` tag matches the value of the principal's `project` tag.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowObjectOperations",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "s3express:CreateSession",
      "Resource": "arn:aws::s3express:region:111122223333:access-point/my-access-
point",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        }
      }
```

```
    }
  ]
}
```

1.3 - IAM policy to modify tags on existing resources maintaining tagging governence

In this IAM policy, IAM principals (users or roles) can modify tags on an access point only if the value of the access point's `project` tag matches the value of the principal's `project` tag. Only the four tags `project`, `environment`, `owner`, and `cost-center` specified in the `aws:TagKeys` condition keys are permitted for these access points. This helps enforce tag governance, prevents unauthorized tag modifications, and keeps the tagging schema consistent across your access points.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceTaggingRulesOnModification",
      "Effect": "Allow",
      "Action": [
        "s3express:TagResource"
      ],
      "Resource": "arn:aws::s3express:region:111122223333:accesspoint/my-access-point",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "project",
            "environment",
            "owner",
            "cost-center"
          ]
        }
      }
    }
  ]
}
```

1.4 - Using the s3express:AccessPointTag condition key

In this IAM policy, the condition statement allows access to the bucket's data only if the access point used to access the bucket has the tag key `Environment` and tag value `Production`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToSpecificAccessPoint",
      "Effect": "Allow",
      "Action": "*",
      "Resource": "arn:aws::s3express:region:111122223333:accesspoint/my-access-point",
      "Condition": {
        "StringEquals": {
          "s3express:AccessPointTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Working with tags for access points for directory buckets

You can add or manage tags for access points for directory buckets using the Amazon S3 Console, the AWS Command Line Interface (CLI), the AWS SDKs, or using the S3 APIs: TagResource, UntagResource, and ListTagsForResource. For more information, see:

Topics

- Creating access points for directory buckets with tags
- Adding a tag to an access point for directory buckets
- Viewing the tags of an access point for directory buckets
- Deleting a tag from an access point for directory buckets

Creating access points for directory buckets with tags

You can tag Amazon S3 Access Points for directory buckets when you create them. For additional information, see Using tags with S3 Access Points for directory buckets.

Permissions

To create an access point for directory buckets with tags, you must have the following permissions:

- `s3express:CreateAccessPoint`
- `s3express:TagResource`

Troubleshooting errors

If you encounter an error when attempting to create an access point for directory buckets with tags, you can do the following:

- Verify that you have the required Permissions to create the access point for directory buckets and add a tag to it.
- Check your IAM user policy for any attribute-based access control (ABAC) conditions. You may be required to label your access points for directory buckets only with specific tag keys and values. For more information, see Using tags for attribute-based access control (ABAC).

Steps

You can create an access point for directory buckets with tags applied by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To create an access point for directory buckets with tags using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points (Directory Buckets)**.
3. Choose **create access point** to create a new access point.
4. Enter a name for the access point. For more information, see Access points for directory buckets naming rules, restrictions, and limitations.
5. On the **Create access point** page, **Tags** is an option when creating a new access point.
6. Choose **Add new Tag** to open the Tags editor and enter a tag key-value pair. The tag key is required, but the value is optional.
7. To add another tag, select **Add new Tag** again. You can enter up to 50 tag key-value pairs.

8. After you complete specifying the options for your new access point, choose **Create access point**.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to create an access point with tags by using the AWS SDK for Java 2.x. To use the command replace the *user input placeholders* with your own information.

```
CreateAccessPointRequest createAccessPointRequest =
 CreateAccessPointRequest.builder()
                .accountId(111122223333)
                .name(my-access-point)
                .bucket(amzn-s3-demo-bucket--zone-id--x-s3)

 .tags(Collections.singletonList(Tag.builder().key("key1").value("value1").build()))
                .build();
 awss3Control.createAccessPoint(createAccessPointRequest);
```

Using the REST API

For information about the Amazon S3 REST API support for creating a directory bucket with tags, see the following section in the *Amazon Simple Storage Service API Reference*:

- CreateBucket

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to create an access point for directory buckets with tags by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

When you create an access point for directory buckets you must provide configuration details and use the following naming convention: *my-access-point*

Request:

```
aws s3control create-access-point \
--account-id 111122223333 \
--name my-access-point \
--bucket amzn-s3-demo-bucket--zone-id--x-s3 \
--profile personal \
--tags Key=key1,Value=value1 Key=MyKey2,Value=value2 \
--region region
```

Adding a tag to an access point for directory buckets

You can add tags to Amazon S3 Access Points for directory buckets and modify these tags. For additional information, see Using tags with S3 Access Points for directory buckets.

Permissions

To add a tag to an access point for directory buckets, you must have the following permission:

- `s3express:TagResource`

Troubleshooting errors

If you encounter an error when attempting to add a tag to an access point for directory buckets, you can do the following:

- Verify that you have the required Permissions to add a tag to an access point for directory buckets.
- If you attempted to add a tag key that starts with the AWS reserved prefix `aws:`, change the tag key and try again.

Steps

You can add tags to access points for directory buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To add tags to an access point for directory buckets using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points (Directory Buckets)**.

3. Choose the access point name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and choose **Add new Tag**.
6. This opens the **Add Tags** page. You can enter up to 50 tag key value pairs.
7. If you add a new tag with the same key name as an existing tag, the value of the new tag overrides the value of the existing tag.
8. You can also edit the values of existing tags on this page.
9. After you have added the tag(s), choose **Save changes**.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to add tags to an access point for directory buckets by using the AWS SDK for Java 2.x. To use the command replace the *user input placeholders* with your own information.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;
import software.amazon.awssdk.services.s3control.model.TagResourceResponse;

public class TagResourceExample {
    public static void tagResourceExample() {
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                .resourceArn("arn:aws::s3:region:111122223333:accesspoint/my-access-
point/*")
                .accountId("111122223333")
                .tags(Tag.builder().key("key1").value("value1").build())
                .build();

        TagResourceResponse response = s3Control.tagResource(tagResourceRequest);
        System.out.println("Status code (should be 204):");
        System.out.println(response.sdkHttpResponse().statusCode());
    }
```

```
}
```

Using the REST API

For information about the Amazon S3 REST API support for adding tags to an access point for directory buckets, see the following section in the *Amazon Simple Storage Service API Reference*:

- TagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to add tags to a directory bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws s3control tag-resource \
--account-id 111122223333 \
--resource-arn arn:aws::s3express:region:444455556666:bucket/prefix--use1-az4--x-s3 \
--tags "Key=key1,Value=value1"
```

Response:

```
{
  "ResponseMetadata": {
      "RequestId": "EXAMPLE123456789",
      "HTTPStatusCode": 200,
      "HTTPHeaders": {
          "date": "Wed, 19 Jun 2025 10:30:00 GMT",
          "content-length": "0"
      },
      "RetryAttempts": 0
  }
}
```

Viewing the tags of an access point for directory buckets

You can view or list tags applied to Amazon S3 Access Points for directory buckets. For additional information, see Using tags with S3 directory buckets.

Permissions

To view tags applied to an access point, you must have the following permission:

- `s3express:ListTagsForResource`

Troubleshooting errors

If you encounter an error when attempting to list or view the tags of an access point for directory buckets, you can do the following:

- Verify that you have the required Permissions to view or list the tags of the access point for directory buckets.

Steps

You can view tags applied to access points for directory buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To view tags applied to an access point for directory buckets using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points (Directory Buckets)**.
3. Choose the access point name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section to view all of the tags applied to the access point for directory buckets.
6. The **Tags** section shows the **User-defined tags** by default. You can select the **AWS-generated tags** tab to view tags applied to your access point by AWS services.

Using the AWS SDKs

This section provides an example of how to view tags applied to an access point for directory buckets by using the AWS SDKs.

SDK for Java 2.x

This example shows you how to view tags applied to an access point for directory buckets by using the AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceRequest;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceResponse;

public class ListTagsForResourceExample {
    public static void listTagsForResourceExample() {
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                .resourceArn("arn:aws::s3:us-west-2:111122223333:accesspoint/my-
access-point/*")
                .accountId("111122223333")
                .build();
        ListTagsForResourceResponse response =
 s3Control.listTagsForResource(listTagsForResourceRequest);
        System.out.println("Tags on my resource:");
        System.out.println(response.toString());
    }
}
```

Using the REST API

For information about the Amazon S3 REST API support for viewing the tags applied to a directory bucket, see the following section in the *Amazon Simple Storage Service API Reference*:

- ListTagsforResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to view tags applied to an access point for directory buckets. To use the command replace the *user input placeholders* with your own information.

Request:

```
aws s3control list-tags-for-resource \
--account-id 111122223333 \
--resource-arn arn:aws::s3express:region:444455556666:bucket/prefix--use1-az4--x-s3 \
```

Response - tags present:

```
{
  "Tags": [
      {
          "Key": "MyKey1",
          "Value": "MyValue1"
      },
      {
          "Key": "MyKey2",
          "Value": "MyValue2"
      },
      {
          "Key": "MyKey3",
          "Value": "MyValue3"
      }
  ]
}
```

Response - no tags present:

```
{
  "Tags": []
}
```

Deleting a tag from an access point for directory buckets

You can remove tags from Access Points for directory buckets. For additional information, see Using tags with S3 Access Points for directory buckets.

> **Note**
>
> If you delete a tag and later learn that it was being used to track costs or for access control, you can add the tag back to the access point for directory buckets.

Permissions

To delete a tag from an access point for directory buckets, you must have the following permission:

- `s3express:UntagResource`

Troubleshooting errors

If you encounter an error when attempting to delete a tag from an access point for directory buckets, you can do the following:

- Verify that you have the required Permissions to delete a tag from an access point for directory buckets.

Steps

You can delete tags from access points for directory buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To delete tags from an access point for directory buckets using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points (Directory Buckets)**.
3. Choose the access point name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and select the checkbox next to the tag or tags that you would like to delete.
6. Choose **Delete**.
7. The **Delete user-defined tags** pop-up appears and asks you to confirm the deletion of the tag or tags you selected.

8. Choose **Delete** to confirm.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to delete tags from a directory bucket by using the AWS SDK for Java 2.x. To use the command replace the *user input placeholders* with your own information.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceRequest;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceResponse;

public class ListTagsForResourceExample {
    public static void listTagsForResourceExample() {
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        UntagResourceRequest untagResourceRequest = UntagResourceRequest.builder()
                .resourceArn("arn:aws::s3:region:111122223333:accesspoint/my-access-
point/*")
                .accountId("111122223333")
                .tagKeys("key1")
                .build();

        UntagResourceResponse response =
 s3Control.untagResource(untagResourceRequest);
        System.out.println("Status code (should be 204):");
        System.out.println(response.sdkHttpResponse().statusCode());
    }
}
```

Using the REST API

For information about the Amazon S3 REST API support for deleting tags from an access point, see the following section in the *Amazon Simple Storage Service API Reference*:

- UnTagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to delete tags from an access point by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws s3control untag-resource \
--account-id 111122223333 \
--resource-arn arn:aws::s3:region:111122223333:accesspoint/my-access-point/* \
--tag-keys "key1" "key2"
```

Response:

```
{
  "ResponseMetadata": {
    "RequestId": "EXAMPLE123456789",
    "HTTPStatusCode": 204,
    "HTTPHeaders": {
        "date": "Wed, 19 Jun 2025 10:30:00 GMT",
        "content-length": "0"
    },
    "RetryAttempts": 0
  }
}
```

Logging with AWS CloudTrail for directory buckets

Amazon S3 is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures all API calls for Amazon S3 as events. Using the information collected by CloudTrail, you can determine the request that was made to Amazon S3, the IP address from which the request was made, when it was made, and additional details. When a supported event activity occurs in Amazon S3, that activity is recorded in a CloudTrail event. You can use AWS CloudTrail trail to log management events and data events for directory buckets. For more information, see Amazon S3 CloudTrail events and What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

CloudTrail management events for directory buckets

By default, CloudTrail logs bucket-level actions for directory buckets as management events. The `eventsource` for CloudTrail management events for directory buckets is `s3express.amazonaws.com`. When you set up your AWS account, CloudTrail management events are enabled by default. The following Regional endpoint API operations (bucket-level, or control plane, API operations) are logged to CloudTrail.

- `CreateBucket`
- `DeleteBucket`
- `DeleteBucketPolicy`
- `PutBucketPolicy`
- `GetBucketPolicy`
- `ListDirectoryBuckets`
- `ListMultipartUploads`
- `GetBucketEncryption`
- `PutBucketEncryption`
- `DeleteBucketEncryption`

> **Note**
>
> `ListMultipartUploads` is a Zonal endpoint API operation. However, this API operation is logged to CloudTrail as a management event. For more information, see `ListMultipartUploads` in the *Amazon Simple Storage Service API Reference*.

For more information on CloudTrail management events, see Logging management events in the *AWS CloudTrail User Guide.*

CloudTrail data events for directory buckets

Data events provide information about the resource operations performed on or in a resource (for example, reading or writing to an Amazon S3 object). These are also known as data plane operations. Data events are often high-volume activities. By default, CloudTrail trails don't log data events, but you can configure trails to log data events for objects stored in general purpose buckets

and directory buckets. For more information, see Enable logging for objects in a bucket using the console .

When you log data events for a trail in CloudTrail, you can choose to use advanced event selectors or basic event selectors. To log data events for objects stored in directory buckets, you must use advanced event selectors. When configuring advanced resource selectors, you will choose or specify the resource type which is `AWS::S3Express::Object`.

The following Zonal endpoint API operations (object-level , or. data plane, API operations) are logged to CloudTrail.

- `AbortMultipartUpload`
- `CompleteMultipartUpload`
- `CreateSession`
- `CopyObject`
- `CreateMultipartUpload`
- `DeleteObject`
- `DeleteObjects`
- `GetObject`
- `GetObjectAttributes`
- `HeadBucket`
- `HeadObject`
- `ListObjectsV2`
- `ListParts`
- `PutObject`
- `RenameObject`
- `UploadPart`
- `UploadPartCopy`

For more information on CloudTrail data events, see Logging data events in the *AWS CloudTrail User Guide*.

For additional information about CloudTrail events for directory buckets, see the following topics:

Topics

- CloudTrail log file examples for directory buckets

CloudTrail log file examples for directory buckets

A CloudTrail log file includes information about the requested API operation, the date and time of the operation, request parameters, and so on. This topic features examples for CloudTrail data events and management events for directory buckets.

Topics

- CloudTrail data event log file examples for directory buckets

CloudTrail data event log file examples for directory buckets

The following example shows a CloudTrail log file example that demonstrates CreateSession.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
      "type": "AssumedRole",
      "principalId": "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName",
      "arn": "arn:aws:sts::111122223333assumed-role/RoleToBeAssumed/MySessionName",
      "accountId": "111122223333",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
      "sessionContext": {
        "sessionIssuer": {
          "type": "Role",
          "principalId": "AROAIDPPEZS35WEXAMPLE",
          "arn": "arn:aws:iam::111122223333:role/RoleToBeAssumed",
          "accountId": "111122223333",
          "userName":"RoleToBeAssumed
        },

        "attributes": {
          "creationDate": "2024-07-02T00:21:16Z",
        "mfaAuthenticated": "false"
        }
      }
    },
    "eventTime": "2024-07-02T00:22:11Z",
    "eventSource": "s3express.amazonaws.com",
    "eventName": "CreateSession",
```

```
        "awsRegion": "us-west-2",
        "sourceIPAddress": "72.21.198.68",
        "userAgent": "aws-sdk-java/2.20.160-SNAPSHOT
 Linux/5.10.216-225.855.amzn2.x86_64 OpenJDK_64-Bit_Server_VM/11.0.23+9-LTS
 Java/11.0.23 vendor/Amazon.com_Inc. md/internal exec-env/AWS_Lambda_java11 io/sync
 http/Apache cfg/retry-mode/standard",
        "requestParameters": {
          "bucketName": "bucket-base-name--usw2-az1--x-s3".
            "host": "bucket-base-name--usw2-az1--x-s3.s3express-usw2-az1.us-
west-2.amazonaws.com",
            "x-amz-create-session-mode": "ReadWrite"
        },
        "responseElements": {
            "credentials": {
                "accessKeyId": "AKIAI44QH8DHBEXAMPLE"
                "expiration": ""Mar 20, 2024, 11:16:09 PM",
                "sessionToken": "<session token string>"
           },
        },
        "additionalEventData": {
            "SignatureVersion": "SigV4",
            "cipherSuite": "TLS_AES_128_GCM_SHA256",
            "bytesTransferredIn": 0,
            "AuthenticationMethod": "AuthHeader",
            "xAmzId2": "q6xhNJYmhg",
            "bytesTransferredOut": 1815,
            "availabilityZone": "usw2-az1"
          },
          "requestID": "28d2faaf-3319-4649-998d-EXAMPLE72818",
          "eventID": "694d604a-d190-4470-8dd1-EXAMPLEe20c1",
          "readOnly": true,
          "resources": [
            {
              "type": "AWS::S3Express::Object",
              "ARNPrefix": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--
usw2-az1--x-s3"
            },
            {
              "accountId": "111122223333"
              "type": "AWS::S3Express::DirectoryBucket",
              "ARN": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--usw2-
az1--x-s3"
             }
           ],
```

```
            "eventType": "AwsApiCall",
            "managementEvent": false,
            "recipientAccountId": "111122223333",
            "eventCategory": "Data",
            "tlsDetails": {
              "tlsVersion": "TLSv1.3",
              "cipherSuite": "TLS_AES_128_GCM_SHA256",
              "clientProvidedHostHeader": "bucket-base-name--usw2-az1--x-s3.s3express-
usw2-az1.us-west-2.amazonaws.com"
            }
          }
```

To use Zonal endpoint API operations (object-level, or data plane, operations), you can use the `CreateSession` API operation to create and manage sessions that are optimized for low-latency authorization of data requests. You can also use `CreateSession` to reduce the amount of logging. To identify which Zonal API operations were performed during a session, you can match the `accessKeyId` under the `responseElements` in your `CreateSession` log file to the `accessKeyId` in the log file of other Zonal API operations. For more information, see [`CreateSession` authorization].

The following example shows a CloudTrail log file example that demonstrates the [`GetObject`] API operation that was authenticated by `CreateSession`.

```
    {
        "eventVersion": "1.09",
        "userIdentity": {
          "type": "AssumedRole",
          "principalId": "AROAIDPPEZS35WEXAMPLE:AssumedRoleSessionName",
          "arn": "arn:aws:sts::111122223333assumed-role/RoleToBeAssumed/MySessionName",
          "accountId": "111122223333",
          "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
          "sessionContext": {
            "attributes": {
              "creationDate": "2024-07-02T00:21:49Z"
            }
          }
        },
        "eventTime": "2024-07-02T00:22:01Z",
        "eventSource": "s3express.amazonaws.com",
        "eventName": "GetObject",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "72.21.198.68",
```

```
        "userAgent": "aws-sdk-java/2.25.66 Linux/5.10.216-225.855.amzn2.x86_64
 OpenJDK_64-Bit_Server_VM/17.0.11+9-LTS Java/17.0.11 vendor/Amazon.com_Inc. md/internal
 exec-env/AWS_Lambda_java17 io/sync http/Apache cfg/retry-mode/legacy",
        "requestParameters": {
          "bucketName": "bucket-base-name--usw2-az1--x-s3",
          "x-amz-checksum-mode": "ENABLED",
          "Host": "bucket-base-name--usw2-az1--x-s3.s3express-usw2-az1.us-
west-2.amazonaws.com",
          "key": "test-get-obj-with-checksum"
        },
        "responseElements": null,
        "additionalEventData": {
          "SignatureVersion": "Sigv4",
          "CipherSuite": "TLS_AES_128_GCM_SHA256",
          "bytesTransferredIn": 0,
          "AuthenticationMethod": "AuthHeader",
          "x-amz-id-2": "oOy6w8K7LFsyFN",
          "bytesTransferredOut": 9,
          "availabilityZone": "usw2-az1",
          "sessionModeApplied": "ReadWrite"
         },
          "requestID": "28d2faaf-3319-4649-998d-EXAMPLE72818",
          "eventID": "694d604a-d190-4470-8dd1-EXAMPLEe20c1",
          "readOnly": true,
          "resources": [
            {
              "type": "AWS::S3Express::Object",
              "ARNPrefix": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--
usw2-az1--x-s3"
            },
            {
              "accountId": "111122223333",
              "type": "AWS::S3Express::DirectoryBucket",
              "ARN": "arn:aws:s3express:us-west-2:111122223333:bucket-base-name--usw2-
az1--x-s3"
             }
           ],
           "eventType": "AwsApiCall",
           "managementEvent": false,
           "recipientAccountId": "111122223333",
           "eventCategory": "Data",
           "tlsDetails": {
             "tlsVersion": "TLSv1.3",
             "cipherSuite": "TLS_AES_128_GCM_SHA256",
```

```
            "clientProvidedHostHeader": "bucket-base-name--usw2-az1--x-s3.s3express-
usw2-az1.us-west-2.amazonaws.com"
          }
        }
```

In the `GetObject` log file example above, the `accessKeyId`(AKIAI44QH8DHBEXAMPLE) matches the `accessKeyId` under the `responseElements` in the CreateSession log file example. The matching `accessKeyId` indicates the session in which `GetObject` operation was performed.

The following example shows a CloudTrail log entry that demonstrates a `DeleteObjects` action on a directory bucket, invoked by S3 Lifecycle. For more information, see Working with S3 Lifecycle for directory buckets.

```
eventVersion:"1.09",
  userIdentity:{

    type:"AWSService",
    invokedBy:"lifecycle.s3.amazonaws.com"
  },
  eventTime:"2024-09-11T00:55:54Z",
  eventSource:"s3express.amazonaws.com",
  eventName:"DeleteObjects",
  awsRegion:"us-east-2",
  sourceIPAddress:"lifecycle.s3.amazonaws.com",
  userAgent:"gamma.lifecycle.s3.amazonaws.com",
  requestParameters:{

    bucketName:"amzn-s3-demo-bucket--use2-az2--x-s3",
    'x-amz-expected-bucket-owner':"637423581905",
    Host:"amzn-s3-demo-bucket--use2-az2--x-s3.gamma.use2-az2.express.s3.aws.dev",
    delete:"",
    'x-amz-sdk-checksum-algorithm':"CRC32C"
  },
  responseElements:null,
  additionalEventData:{

    SignatureVersion:"Sigv4",
    CipherSuite:"TLS_AES_128_GCM_SHA256",
    bytesTransferredIn:41903,
    AuthenticationMethod:"AuthHeader",
    'x-amz-id-2':"9H5YWZY0",
    bytesTransferredOut:35316,
    availabilityZone:"use2-az2",
```

```
    sessionModeApplied:"ReadWrite"
  },
  requestID:"011eeadd04000191",
  eventID:"d3d8b116-219d-4ee6-a072-5f9950733c74",
  readOnly:false,
  resources:[

    {

      type:"AWS::S3Express::Object",
      ARNPrefix:"arn:aws:s3express:us-east-2:637423581905:bucket/amzn-s3-demo-bucket--
use2-az2--x-s3/"
    },
    {

      accountId:"637423581905",
      type:"AWS::S3Express::DirectoryBucket",
      ARN:"arn:aws:s3express:us-east-2:637423581905:bucket/amzn-s3-demo-bucket--use2-
az2--x-s3"
    }
  ],
  eventType:"AwsApiCall",
  managementEvent:false,
  recipientAccountId:"637423581905",
  sharedEventID:"59f877ac-1dd9-415d-b315-9bb8133289ce",
  eventCategory:"Data"
}
```

The following example shows a CloudTrail log entry that demonstrates an `Access Denied` request on a `CreateSession` action invoked by S3 Lifecycle. For more information, see CreateSession.

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "gamma.lifecycle.s3.amazonaws.com"
    },
    "eventTime": "2024-09-11T18:13:08Z",
    "eventSource": "s3express.amazonaws.com",
    "eventName": "CreateSession",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "gamma.lifecycle.s3.amazonaws.com",
```

```
    "userAgent": "gamma.lifecycle.s3.amazonaws.com",
    "errorCode": "AccessDenied",
    "errorMessage": "Access Denied",
    "requestParameters": {
        "bucketName": "amzn-s3-demo-bucket--use2-az2--x-s3",
        "Host": "amzn-s3-demo-bucket--use2-az2--x-s3.gamma.use2-
az2.express.s3.aws.dev",
        "x-amz-create-session-mode": "ReadWrite",
        "x-amz-server-side-encryption": "AES256"
    },
    "responseElements": null,
    "additionalEventData": {
        "SignatureVersion": "Sigv4",
        "CipherSuite": "TLS_AES_128_GCM_SHA256",
        "bytesTransferredIn": 0,
        "AuthenticationMethod": "AuthHeader",
        "x-amz-id-2": "zuDDC1VNbC4LoNwUIc5",
        "bytesTransferredOut": 210,
        "availabilityZone": "use2-az2"
    },
    "requestID": "010932f174000191e24a0",
    "eventID": "dce7cc46-4cd3-46c0-9a47-d1b8b70e301c",
    "readOnly": true,
    "resources": [{
            "type": "AWS::S3Express::Object",
            "ARNPrefix": "arn:aws:s3express:us-east-2:637423581905:bucket/amzn-s3-demo-
bucket--use2-az2--x-s3/"
        },
        {
            "accountId": "637423581905",
            "type": "AWS::S3Express::DirectoryBucket",
            "ARN": "arn:aws:s3express:us-east-2:637423581905:bucket/amzn-s3-demo-
bucket--use2-az2--x-s3"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "637423581905",
    "sharedEventID": "da96b5bd-6066-4a8d-ad8d-f7f427ca7d58",
    "eventCategory": "Data"
}
```

Monitoring metrics with Amazon CloudWatch for directory buckets

Amazon CloudWatch metrics for directory buckets can help you understand and improve the performance of applications that use directory buckets. There are several sets of CloudWatch metrics that you can use with directory buckets for the S3 Express One Zone storage class and the S3 One Zone-Infrequent Access (S3 One Zone-IA; Z-IA) storage class in a local zone.

Daily storage metrics

Monitor the amount of data stored in directory buckets, including total size in bytes and number of objects. These storage metrics for S3 Express One Zone are reported once per day and are provided to all customers at no additional cost.

Request metrics

Monitor directory bucket requests to quickly identify and act on operational issues. The metrics are available at 1-minute intervals after some latency for processing. These CloudWatch metrics are billed at the same rate as the Amazon CloudWatch custom metrics. For information about CloudWatch pricing, see Amazon CloudWatch pricing. To learn how to opt in to getting these metrics, see Configuring request metrics for directory buckets.

When enabled, request metrics are reported for all object operations. By default, these 1-minute metrics are available at the directory bucket level. You can also define a filter for the metrics using a shared directory or access point:

- **Access point** – Access points are named network endpoints that are attached to directory buckets and simplify managing data access at scale for shared datasets in S3. With the access point filter, you can gain insights into your access point usage. For more information about access points for directory buckets, see Managing access to shared datasets in directory buckets with access points.
- **Directory** – directory buckets use actual directories to organize objects hierarchically. A directory enables you to group similar objects together in a directory bucket. If you filter by directory, objects that are stored in the same directory are included in the metrics configuration.

To align these metrics to specific business applications, workflows, or internal organizations, you can filter on a shared directory or access point.

All CloudWatch statistics are retained for a period of 15 months so that you can access historical information and gain a better perspective on how your web application or service is performing. For more information about CloudWatch, see What is Amazon CloudWatch? in the *Amazon CloudWatch User Guide*. You may need some additional configurations to your CloudWatch alarms, depending on your use cases. For example, you can use metric math expression to create an alarm. For more information, see Use CloudWatch metrics, Use metric math, Using Amazon CloudWatch alarms, and Create a CloudWatch alarm based on a metric math expression in the *Amazon CloudWatch User Guide*.

Best-effort CloudWatch metrics delivery

CloudWatch metrics are delivered on a best-effort basis. Most requests for an Amazon S3 object that have request metrics result in a data point being sent to CloudWatch.

The completeness and timeliness of metrics are not guaranteed. The data point for a particular request might be returned with a timestamp that is later than when the request was actually processed. The data point for a minute might be delayed before being available through CloudWatch, or it might not be delivered at all. CloudWatch request metrics give you an idea of the nature of traffic against your bucket in near-real time. It is not meant to be a complete accounting of all requests.

It follows from the best-effort nature of this feature that the reports available at the Billing & Cost Management Dashboard might include one or more access requests that do not appear in the bucket metrics.

For more information, see the following topics.

Topics

- Metrics and dimensions for directory buckets
- Configuring request metrics for directory buckets

Metrics and dimensions for directory buckets

The metrics and dimensions that S3 Express One Zone send to Amazon CloudWatch are listed in the following tables.

Best-effort CloudWatch metrics delivery

CloudWatch metrics are delivered on a best-effort basis. Most requests for an Amazon S3 object that have request metrics result in a data point being sent to CloudWatch.

The completeness and timeliness of metrics are not guaranteed. The data point for a particular request might be returned with a timestamp that is later than when the request was actually processed. The data point for a minute might be delayed before being available through CloudWatch, or it might not be delivered at all. CloudWatch request metrics give you an idea of the nature of traffic against your bucket in near-real time. It is not meant to be a complete accounting of all requests.

It follows from the best-effort nature of this feature that the reports available at the Billing & Cost Management Dashboard might include one or more access requests that do not appear in the bucket metrics.

Topics

- Amazon S3 daily storage metrics for directory buckets in CloudWatch
- Amazon S3 request metrics for directory buckets in CloudWatch
- Amazon S3 dimensions for directory buckets

Amazon S3 daily storage metrics for directory buckets in CloudWatch

The `AWS/S3` namespace includes the following daily storage metrics for directory buckets.

Metric	Description
`BucketSizeBytes`	The amount of data in bytes that is stored in a directory bucket. This value is calculated by summing the size of all objects and metadata (such as bucket names) in the bucket, including the size of all parts for all incomplete multipart uploads to the bucket. Units: Bytes Valid statistics: Average
`NumberOfObjects`	The total number of objects stored in a directory bucket. This value is calculated by counting all objects in the bucket and doesn't include incomplete multipart uploads to the bucket. Units: Count

Metric	Description
	Valid statistics: Average

Amazon S3 request metrics for directory buckets in CloudWatch

The `AWS/S3` namespace includes the following request metrics for directory buckets.

Metric	Description
`AllRequests`	The total number of HTTP requests made to a directory bucket, regardless of type. If you're using a metrics configuration with a filter, then this metric returns only the HTTP requests that meet the filter's requirements. Units: Count Valid statistics: Sum
`GetRequests`	The number of HTTP `GET` requests made for objects in a directory bucket. This doesn't include list operations. This metric is incremented for the source of each `CopyObject` request. Units: Count Valid statistics: Sum **Note** Paginated list-oriented requests, such as ListMultipartUploads, ListParts, and others, are not included in this metric.
`PutRequests`	The number of HTTP `PUT` requests made for objects in a directory bucket. This metric is incremented for the destination of each `CopyObject` request. Units: Count

Metric	Description
	Valid statistics: Sum
`DeleteRequests`	The number of HTTP `DELETE` requests made for objects in a directory bucket. This metric also includes DeleteObjects requests. This metric shows the number of requests made, not the number of objects deleted. Units: Count Valid statistics: Sum
`HeadRequests`	The number of HTTP `HEAD` requests made to a directory bucket. Units: Count Valid statistics: Sum
`PostRequests`	The number of HTTP `POST` requests made to a directory bucket. Units: Count Valid statistics: Sum **Note** DeleteObjects requests are not included in this metric.
`ListRequests`	The number of HTTP requests that list the contents of a directory bucket. Units: Count Valid statistics: Sum

Metric	Description
`BytesDownloaded`	The number of bytes downloaded for requests made to a directory bucket, where the response includes a body. Units: Bytes Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9
`BytesUploaded`	The number of bytes uploaded for requests made to a directory bucket, where the request includes a body. Units: Bytes Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9
`4xxErrors`	The number of HTTP 4*xx* client error status code requests made to a directory bucket with a value of either 0 or 1. The Average statistic shows the error rate, and the Sum statistic shows the count of that type of error, during each period. Units: Count Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count
`5xxErrors`	The number of HTTP 5*xx* server error status code requests made to a directory bucket with a value of either 0 or 1. The Average statistic shows the error rate, and the Sum statistic shows the count of that type of error, during each period. Units: Count Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count

Metric	Description
`FirstByteLatency`	The per-request time from the complete request being received by a directory bucket to when the response starts to be returned. Units: Milliseconds Valid statistics: Average, Sum, Min, Max (same as p100), Sample Count, any percentile between p0.0 and p100
`TotalRequestLatency`	The elapsed per-request time from the first byte received to the last byte sent to a directory bucket. This metric includes the time taken to receive the request body and send the response body, which is not included in `FirstByteLatency` . Units: Milliseconds Valid statistics: Average, Sum, Min, Max (same as p100), Sample Count, any percentile between p0.0 and p100

Amazon S3 dimensions for directory buckets

The following dimensions are used to filter Amazon S3 metrics for directory buckets.

Dimension	Description
`BucketName`	This dimension filters the data that you request for the identified directory bucket only.
`FilterId`	This dimension filters metrics configurations that you specify for request metrics on a directory bucket. You set up the metrics configuration filter when you configure request metrics. For more information, see Configuring request metrics for directory buckets.

Configuring request metrics for directory buckets

You can configure request metrics for directory buckets to monitor the operational performance of your S3 Express One Zone storage. Request metrics are available at 1-minute intervals and help you quickly identify and act on operational issues.

You can configure request metrics for directory buckets using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), or the Amazon S3 REST API.

Using the S3 console

You can use the Amazon S3 console to configure request metrics for your directory buckets.

To configure request metrics for a directory bucket

1. Sign in to the and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Directory buckets**.
3. In the buckets list, choose the name of the directory bucket that contains the objects you want request metrics for.
4. Choose the **Metrics** tab.
5. Under **Bucket metrics**, choose **View additional charts**.
6. Choose the **Request metrics** tab.
7. Choose **Create filter**.
8. In the **Filter name** box, enter your filter name.

   Names can only contain letters, numbers, periods, dashes, and underscores. We recommend using the name `EntireBucket` for a filter that applies to all objects.
9. Under **Filter scope**, choose one of the following:

   - **This filter applies to all objects in the bucket** – You don't specify a filter.
   - **Limit the scope of this filter using directory or access point** – You specify a filter. To limit the scope, you can filter by directory or access point.

   You can also define a filter so that the metrics are only collected and reported on a subset of objects in the directory bucket.
10. If you chose to limit the scope of the filter, specify the filter details:

- To filter by **directory**, enter a directory name.
- To filter by **access point**, enter the access point ARN.

11. Choose **Save changes**.
12. On the **Request metrics** tab, under **Filters**, choose the filter that you just created.

    After about 15 minutes, CloudWatch begins tracking these request metrics. You can see them on the **Request metrics** tab. You can see graphs for the metrics on the Amazon S3 or CloudWatch console. Request metrics are billed at the standard CloudWatch rate. For more information, see Amazon CloudWatch pricing.

Using the AWS CLI

You can use the AWS CLI to configure request metrics for your directory buckets.

1. Install and set up the AWS CLI. For instructions, see Installing, updating, and uninstalling the AWS CLI in the *AWS Command Line Interface User Guide*.
2. Open a terminal.
3. Run the following command to add a metrics configuration.

```
aws s3api put-bucket-metrics-configuration --endpoint https://s3express-
control.region-code.amazonaws.com --bucket directory-bucket-name --id metrics-
config-id --metrics-configuration '{"Id":"metrics-config-id"}'
```

Using the REST API

You can use the Amazon S3 REST API to configure request metrics for your directory buckets. When using the REST API with directory buckets, you must use the S3 Express One Zone endpoint with the regional control endpoint in the format: `https://s3express-control.region-code.amazonaws.com`.

For more information about the REST API operations, see the following topics in the *Amazon S3 API Reference*:

- PUT Bucket metrics
- GET Bucket metrics
- DELETE Bucket metrics

- LIST Bucket metrics

Optimizing directory bucket performance

To obtain the best performance when using directory buckets, we recommend the following guidelines.

For more information about best practices for S3 Express One Zone, see Best practices to optimize S3 Express One Zone performance.

Use session-based authentication

Directory buckets support a new session-based authorization mechanism to authenticate and authorize requests to a directory bucket. With session-based authentication, the AWS SDKs automatically use the `CreateSession` API operation to create a temporary session token that can be used for low-latency authorization of data requests to a directory bucket.

The AWS SDKs use the `CreateSession` API operation to request temporary credentials, and then automatically create and refresh tokens for you on your behalf every 5 minutes. To take advantage of the performance benefits of directory buckets, we recommended that you use the AWS SDKs to initiate and manage the `CreateSession` API request. For more information about this session-based model, see Authorizing Zonal endpoint API operations with `CreateSession`.

S3 additional checksum best practices

Directory buckets offer you the option to choose the checksum algorithm that is used to validate your data during upload or download. You can select one of the following Secure Hash Algorithms (SHA) or Cyclic Redundancy Check (CRC) data-integrity check algorithms: CRC32, CRC32C, SHA-1, and SHA-256. MD5-based checksums are not supported with the S3 Express One Zone storage class.

CRC32 is the default checksum used by the AWS SDKs when transmitting data to or from directory buckets. We recommend using CRC32 and CRC32C for the best performance with directory buckets.

Use the latest version of the AWS SDKs and common runtime libraries

Several of the AWS SDKs also provide the AWS Common Runtime (CRT) libraries to further accelerate performance in S3 clients. These SDKs include the AWS SDK for Java 2.x, the AWS

SDK for C++, and the AWS SDK for Python (Boto3). The CRT-based S3 client transfers objects to and from directory buckets with enhanced performance and reliability by automatically using the multipart upload API operation and byte-range fetches to automate horizontally scaling connections.

To achieve the highest performance with the directory buckets, we recommend using the latest version of the AWS SDKs that include the CRT libraries or using the AWS Command Line Interface (AWS CLI).

Developing with directory buckets

After you create your directory bucket, you can then immediately begin very low-latency reads and writes. You can communicate with your directory bucket by using an endpoint connection over a virtual private cloud (VPC), or you can use Zonal and Regional API operations to manage your objects and directory buckets. You can work with directory buckets by using the AWS SDKs, Amazon S3 console, AWS Command Line Interface (AWS CLI), and Amazon S3 REST APIs.

Topics

- Regional and Zonal endpoints for directory buckets
- Working with directory buckets by using the S3 console, AWS CLI, and AWS SDKs
- Directory bucket API operations

Regional and Zonal endpoints for directory buckets

To access the Regional and Zonal endpoints for directory buckets from your virtual private cloud (VPC), you can use gateway VPC endpoints. After you create a gateway endpoint, you can add it as a target in your route table for traffic destined from your VPC to your bucket. There is no additional charge for using gateway endpoints. For more information about how to configure gateway VPC endpoints, see Networking for directory buckets.

Bucket-level (control plane) API operations are available through a Regional endpoint and are referred to as Regional endpoint API operations. Examples of Regional endpoint API operations are `CreateBucket` and `DeleteBucket`.

You use Zonal (object level, or data plane endpoint API operations) to upload and manage your objects. Zonal endpoint API operations are available through a Zonal endpoint. Examples of Zonal API operations are `PutObject` and `CopyObject`.

For more information about Regional and Zonal endpoints for directory buckets in Availability Zones, see Regional and Zonal endpoints for directory buckets in an Availability Zone.

For more information about Regional and Zonal endpoints for directory buckets in Local Zones, see Concepts for directory buckets in Local Zones.

Working with directory buckets by using the S3 console, AWS CLI, and AWS SDKs

You can work with the S3 Express One Zone storage class and directory buckets by using the AWS SDKs, Amazon S3 console, AWS Command Line Interface (AWS CLI), and Amazon S3 REST API.

S3 Console

To get started using the S3 console, follow these steps:

- Creating directory buckets in an Availability Zone
- Emptying a directory bucket
- Deleting a directory bucket

For a full tutorial, see Tutorial: Getting started with S3 Express One Zone.

AWS SDKs

S3 Express One Zone supports the following AWS SDKs:

- AWS SDK for C++
- AWS SDK for Go v2
- AWS SDK for Java 2.x
- AWS SDK for JavaScript v3
- AWS SDK for .NET
- AWS SDK for PHP
- AWS SDK for Python (Boto3)
- AWS SDK for Ruby
- AWS SDK for Kotlin

- AWS SDK for Rust

When you're working with S3 Express One Zone, we recommend using the latest version of the AWS SDKs. The supported AWS SDKs for S3 Express One Zone handle session establishment, refreshment, and termination on your behalf. This means that you can immediately start using API operations after you download and install the AWS SDKs and configure the necessary IAM permissions. For more information, see Authorizing Regional endpoint API operations with IAM.

For information about the AWS SDKs, including how to download and install them, see Tools to Build on AWS.

For AWS SDK examples, see the following:

- Creating directory buckets in an Availability Zone
- Emptying a directory bucket
- Deleting a directory bucket

AWS Command Line Interface (AWS CLI)

You can use the AWS Command Line Interface (AWS CLI) to create directory buckets and use supported Regional and Zonal endpoint API operations for S3 Express One Zone.

To get started with the AWS CLI, see Get started with the AWS CLI in the *AWS CLI Command Reference*.

> **Note**
>
> To use directory buckets with the high-level `aws s3` commands, update your AWS CLI to the latest version. For more information about how to install and configure the AWS CLI, see Install or update the latest version of the AWS CLI in the *AWS CLI Command Reference*.

For AWS CLI examples, see the following:

- Creating directory buckets in an Availability Zone
- Emptying a directory bucket
- Deleting a directory bucket

Directory bucket API operations

To manage directory buckets, you can use Regional (bucket level, or control plane) endpoint API operations. To manage objects in your directory buckets, you can use Zonal (object level, or data plane) endpoint API operations. For more information, see Networking for directory buckets and Endpoints and gateway VPC endpoints.

Regional endpoint API operations

The following Regional endpoint API operations are supported for directory buckets:

- CreateAccessPoint
- CreateBucket
- DeleteAccessPoint
- DeleteAccessPointPolicy
- DeleteAccessPointScope
- DeleteBucket
- DeleteBucketInventoryConfiguration
- DeleteBucketLifecycle
- DeleteBucketPolicy
- GetAccessPoint
- GetAccessPointPolicy
- GetAccessPointScope
- GetBucketInventoryConfiguration
- GetBucketLifecycleConfiguration
- GetBucketPolicy
- ListAccessPointsForDirectoryBuckets
- ListBucketInventoryConfigurations
- ListDirectoryBuckets
- ListTagsForResource
- PutAccessPointPolicy
- PutAccessPointScope

- PutBucketInventoryConfiguration
- PutBucketLifecycleConfiguration
- PutBucketPolicy
- DeleteBucketEncryption
- GetBucketEncryption
- PutBucketEncryption
- TagResource
- UntagResource

Zonal endpoint API operations

The following Zonal endpoint API operations are supported for directory buckets:

- CreateSession
- CopyObject
- DeleteObject
- DeleteObjects
- GetObject
- GetObjectAttributes
- HeadBucket
- HeadObject
- ListObjectsV2
- PutObject
- RenameObject
- AbortMultipartUpload
- CompleteMultiPartUpload
- CreateMultipartUpload
- ListMultipartUploads
- ListParts
- UploadPart

- UploadPartCopy

Using tags with S3 directory buckets

An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 directory buckets. You can tag S3 directory buckets when you create them or manage tags on existing directory buckets. For general information about tags, see Tagging for cost allocation or attribute-based access control (ABAC).

> **Note**
>
> There is no additional charge for using tags on directory buckets beyond the standard S3 API request rates. For more information, see Amazon S3 pricing.

Common ways to use tags with directory buckets

Use tags on your S3 directory buckets for:

1. **Cost allocation** – Track storage costs by bucket tag in AWS Billing and Cost Management. For more information, see Using tags for cost allocation.
2. **Attribute-based access control (ABAC)** – Scale access permissions and grant access to S3 directory buckets based on their tags. For more information, see Using tags for ABAC.

> **Note**
>
> You can use the same tags for both cost allocation and access control.

ABAC for S3 directory buckets

Amazon S3 directory buckets support attribute-based access control (ABAC) using tags. Use tag-based condition keys in your AWS organizations, IAM, and S3 directory bucket policies. For enterprises, ABAC in Amazon S3 supports authorization across multiple AWS accounts.

In your IAM policies, you can control access to S3 directory buckets based on the bucket's tags by using the following global condition keys:

- `aws:ResourceTag/key-name`
  - Use this key to compare the tag key-value pair that you specify in the policy with the key-value pair attached to the resource. For example, you could require that access to a resource is allowed only if the resource has the attached tag key `Dept` with the value `Marketing`. For more information, see Controlling access to AWS resources.
- `aws:RequestTag/key-name`
  - Use this key to compare the tag key-value pair that was passed in the request with the tag pair that you specify in the policy. For example, you could check whether the request includes the tag key `Dept` and that it has the value `Accounting`. For more information, see Controlling access during AWS requests. You can use this condition key to restrict which tag key-value pairs can be passed during the `TagResource` and `CreateBucket` API operations.
- `aws:TagKeys`
  - Use this key to compare the tag keys in a request with the keys that you specify in the policy. We recommend that when you use policies to control access using tags, use the `aws:TagKeys` condition key to define what tag keys are allowed. For example policies and more information, see Controlling access based on tag keys. You can create an S3 directory bucket with tags. To allow tagging during the `CreateBucket` API operation, you must create a policy that includes both the `s3express:TagResource` and `s3express:CreateBucket` actions. You can then use the `aws:TagKeys` condition key to enforce using specific tags in the `CreateBucket` request.
- `s3express:BucketTag/tag-key`
  - Use this condition key to grant permissions to specific data in directory buckets using tags. When accessing a directory bucket by using an access point, this condition key references the tags on the directory bucket both when authorizing against the access point and the directory bucket, while the `aws:ResourceTag/tag-key` will reference the tags only of the resource it is being authorized against.

Example ABAC policies for directory buckets

See the following example ABAC policies for Amazon S3 directory buckets.

1.1 - IAM policy to create or modify buckets with specific tags

In this IAM policy, users or roles with this policy can only create S3 directory buckets if they tag the bucket with the tag key `project` and tag value `Trinity` in the bucket creation request. They can also add or modify tags on existing S3 directory buckets as long as the `TagResource` request

includes the tag key-value pair `project:Trinity`. This policy does not grant read, write, or delete permissions on the buckets or its objects.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateBucketWithTags",
      "Effect": "Allow",
      "Action": [
        "s3express:CreateBucket",
        "s3express:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/project": [
            "Trinity"
          ]
        }
      }
    }
  ]
}
```

1.2 - Bucket policy to restrict operations on the bucket using tags

In this bucket policy, IAM principals (users and roles) can perform operations using the `CreateSession` action on the bucket only if the value of the bucket's `project` tag matches the value of the principal's `project` tag.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowObjectOperations",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "s3express:CreateSession",
      "Resource": "arn:aws::s3express:us-west-2:111122223333:bucket/amzn-s3-demo-
bucket--usw2-az1--x-s3",
```

```
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        }
      }
    }
  ]
}
```

1.3 - IAM policy to modify tags on existing resources maintaining tagging governence

In this IAM policy, IAM principals (users or roles) can modify tags on a bucket only if the value of the bucket's `project` tag matches the value of the principal's `project` tag. Only the four tags `project`, `environment`, `owner`, and `cost-center` specified in the `aws:TagKeys` condition keys are permitted for these directory buckets. This helps enforce tag governance, prevents unauthorized tag modifications, and keeps the tagging schema consistent across your buckets.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceTaggingRulesOnModification",
      "Effect": "Allow",
      "Action": [
        "s3express:TagResource"
      ],
      "Resource": "arn:aws::s3express:us-west-2:111122223333:bucket/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "project",
            "environment",
            "owner",
            "cost-center"
          ]
        }
      }
    }
  ]
```

```
}
```

1.4 - Using the s3express:BucketTag condition key

In this IAM policy, the condition statement allows access to the bucket's data only if the bucket has the tag key `Environment` and tag value `Production`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToSpecificAccessPoint",
      "Effect": "Allow",
      "Action": "*",
      "Resource": "arn:aws::s3express:us-west-2:111122223333:accesspoint/*",
      "Condition": {
        "StringEquals": {
          "s3express:BucketTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Managing tags for directory buckets

You can add or manage tags for S3 directory buckets using the Amazon S3 Console, the AWS Command Line Interface (CLI), the AWS SDKs, or using the S3 APIs: TagResource, UntagResource, and ListTagsForResource. For more information, see:

Topics

- Creating directory buckets with tags
- Adding a tag to a directory bucket
- Viewing directory bucket tags
- Deleting a tag from a directory bucket

Creating directory buckets with tags

You can tag Amazon S3 directory buckets when you create them. There is no additional charge for using tags on directory buckets beyond the standard S3 API request rates. For more information, see Amazon S3 pricing. For more information about tagging directory buckets, see Using tags with S3 directory buckets.

Permissions

To create a directory bucket with tags, you must have the following permissions:

- `s3express:CreateBucket`
- `s3express:TagResource`

Troubleshooting errors

If you encounter an error when attempting to create a directory bucket with tags, you can do the following:

- Verify that you have the required Permissions to create the directory bucket and add a tag to it.
- Check your IAM user policy for any attribute-based access control (ABAC) conditions. You may be required to label your directory buckets only with specific tag keys and values. For more information, see Using tags for attribute-based access control (ABAC).

Steps

You can create a directory bucket with tags applied by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To create a directory bucket with tags using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **directory buckets**.
3. Choose **create bucket** to create a new directory bucket.
4. You can create two types of directory buckets:

Create a directory bucket in an Availability Zone for a high performance workload. For more information, see High performance workloads.

Create a directory bucket in a Local Zone for a data residency workload. For more information, see Data residency workloads.

5. For both types of directory buckets, on the **Create bucket** page, **Tags** is an option when creating a new directory bucket.
6. Enter a name for the bucket. For more information, see Directory bucket naming rules.
7. Choose **Add new Tag** to open the Tags editor and enter a tag key-value pair. The tag key is required, but the value is optional.
8. To add another tag, select **Add new Tag** again. You can enter up to 50 tag key-value pairs.
9. After you complete specifying the options for your new directory bucket, choose **Create bucket**.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to create a directory bucket with tags by using the AWS SDK for Java 2.x. To use the command replace the *`user input placeholders`* with your own information.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.Tag;

public class CreateBucketWithTagsExample {
    public static void createBucketWithTagsExample() {
        S3Client s3 = S3Client.builder().region(Region.US_WEST_2).build();

        CreateBucketConfiguration bucketConfiguration =
 CreateBucketConfiguration.builder()
```

```
                .location(LocationInfo.builder()
                        .type(LocationType.AVAILABILITY_ZONE)
                        .name("usw2-az1").build())
                .bucket(BucketInfo.builder()
                        .type(BucketType.DIRECTORY)
                        .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                        .build())
                .tags(Tag.builder().key("MyTagKey").value("MyTagValue").build())
                .build();

        CreateBucketRequest createBucketRequest = CreateBucketRequest.builder()
                .bucket("amzn-s3-demo-bucket--usw2-az1--x-s3--usw2-az1--x-s3")
                .createBucketConfiguration(bucketConfiguration)
                .build();

        CreateBucketResponse response = s3.createBucket(createBucketRequest);
        System.out.println("Status code (should be 200):");
        System.out.println(response.sdkHttpResponse().statusCode());
    }
}
```

Using the REST API

For information about the Amazon S3 REST API support for creating a directory bucket with tags, see the following section in the *Amazon Simple Storage Service API Reference*:

- CreateBucket

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to create a directory bucket with tags by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

When you create a directory bucket you must provide configuration details and use the following naming convention: *`bucket-base-name`*`--`*`zone-id`*`--x-s3`

Request:

```
aws s3api create-bucket \
```

```
--bucket bucket-base-name--zone-id--x-s3 \
--create-bucket-configuration "Location={Type=AvailabilityZone,Name=zone-
id},Bucket={DataRedundancy=SingleAvailabilityZone,Type=Directory},Tags=[{Key=mykey1,Value=myval
 {Key=mykey2,Value=myvalue2}]"
```

Response:

```
{
  "Location": "http://bucket--use1-az4--x-s3.s3express-use1-az4.us-
east-1.amazonaws.com/"
}
```

Adding a tag to a directory bucket

You can add tags to Amazon S3 directory buckets and modify these tags. There is no additional charge for using tags on directory buckets beyond the standard S3 API request rates. For more information, see Amazon S3 pricing. For more information about tagging directory buckets, see Using tags with S3 directory buckets.

Permissions

To add a tag to a directory bucket, you must have the following permission:

- `s3express:TagResource`

Troubleshooting errors

If you encounter an error when attempting to add a tag to a directory bucket, you can do the following:

- Verify that you have the required Permissions to add a tag to a directory bucket.
- If you attempted to add a tag key that starts with the AWS reserved prefix `aws:`, change the tag key and try again.

Steps

You can add tags to directory buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To add tags to a directory bucket using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **directory buckets**.
3. Choose the bucket name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and choose **Add new Tag**.
6. This opens the **Add Tags** page. You can enter up to 50 tag key value pairs.
7. If you add a new tag with the same key name as an existing tag, the value of the new tag overrides the value of the existing tag.
8. You can also edit the values of existing tags on this page.
9. After you have added the tag(s), choose **Save changes**.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to add tags to a directory bucket by using the AWS SDK for Java 2.x. To use the command replace the *user input placeholders* with your own information.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;
import software.amazon.awssdk.services.s3control.model.TagResourceResponse;

public class TagResourceExample {
    public static void tagResourceExample() {
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                .resourceArn("arn:aws::s3express:us-west-2:111122223333:bucket/my-
directory-bucket--usw2-az1--x-s3")
                .accountId("111122223333")
                .tags(Tag.builder().key("MyTagKey").value("MyTagValue").build())
```

```
                .build();

        TagResourceResponse response = s3Control.tagResource(tagResourceRequest);
        System.out.println("Status code (should be 204):");
        System.out.println(response.sdkHttpResponse().statusCode());
    }
}
```

Using the REST API

For information about the Amazon S3 REST API support for adding tags to a directory bucket, see the following section in the *Amazon Simple Storage Service API Reference*:

- TagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to add tags to a directory bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws s3control tag-resource \
--account-id 111122223333 \
--resource-arn arn:aws::s3express:us-east-1:444455556666:bucket/prefix--use1-az4--x-s3
 \
--tags "Key=mykey,Value=myvalue"
```

Response:

```
{
  "ResponseMetadata": {
      "RequestId": "EXAMPLE123456789",
      "HTTPStatusCode": 200,
      "HTTPHeaders": {
          "date": "Wed, 19 Jun 2025 10:30:00 GMT",
          "content-length": "0"
      },
      "RetryAttempts": 0
```

```
  }
}
```

Viewing directory bucket tags

You can view or list tags applied to S3 directory buckets. For more information about tags, see Using tags with S3 directory buckets.

Permissions

To view tags applied to a directory bucket, you must have the following permission:

- `s3express:ListTagsForResource`

Troubleshooting errors

If you encounter an error when attempting to list or view the tags of a directory bucket, you can do the following:

- Verify that you have the required Permissions to view or list the tags of the directory bucket.

Steps

You can view tags applied to directory buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To view tags applied to a directory bucket using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **directory buckets**.
3. Choose the bucket name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section to view all of the tags applied to the directory bucket.
6. The **Tags** section shows the **User-defined tags** by default. You can select the **AWS-generated tags** tab to view tags applied to your directory bucket by AWS services.

Using the AWS SDKs

This section provides an example of how to view tags applied to a directory bucket by using the AWS SDKs.

SDK for Java 2.x

This example shows you how to view tags applied to a directory bucket by using the AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceRequest;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceResponse;

public class ListTagsForResourceExample {
    public static void listTagsForResourceExample() {
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                .resourceArn("arn:aws::s3express:us-west-2:111122223333:bucket/my-
directory-bucket--usw2-az1--x-s3")
                .accountId("111122223333")
                .build();

        ListTagsForResourceResponse response =
 s3Control.listTagsForResource(listTagsForResourceRequest);
        System.out.println("Tags on my resource:");
        System.out.println(response.toString());
    }
}
```

Using the REST API

For information about the Amazon S3 REST API support for viewing the tags applied to a directory bucket, see the following section in the *Amazon Simple Storage Service API Reference*:

- ListTagsforResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to view tags applied to a directory bucket. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws s3control list-tags-for-resource \
--account-id 111122223333 \
--resource-arn arn:aws::s3express:us-east-1:444455556666:bucket/prefix--use1-az4--x-s3
 \
```

Response - tags present:

```
{
  "Tags": [
      {
          "Key": "MyKey1",
          "Value": "MyValue1"
      },
      {
          "Key": "MyKey2",
          "Value": "MyValue2"
      },
      {
          "Key": "MyKey3",
          "Value": "MyValue3"
      }
  ]
}
```

Response - no tags present:

```
{
  "Tags": []
}
```

Deleting a tag from a directory bucket

You can remove tags from S3 directory buckets. An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 directory buckets. For more information about tags, see Using tags with S3 directory buckets.

> **Note**
>
> If you delete a tag and later learn that it was being used to track costs or for access control, you can add the tag back to the directory bucket.

Permissions

To delete a tag from a directory bucket, you must have the following permission:

- `s3express:UntagResource`

Troubleshooting errors

If you encounter an error when attempting to delete a tag from a directory bucket, you can do the following:

- Verify that you have the required Permissions to delete a tag from a directory bucket.

Steps

You can delete tags from directory buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To delete tags from a directory bucket using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **directory buckets**.
3. Choose the bucket name.
4. Choose the **Properties** tab.

5. Scroll to the **Tags** section and select the checkbox next to the tag or tags that you would like to delete.
6. Choose **Delete**.
7. The **Delete user-defined tags** pop-up appears and asks you to confirm the deletion of the tag or tags you selected.
8. Choose **Delete** to confirm.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to delete tags from a directory bucket by using the AWS SDK for Java 2.x. To use the command replace the *`user input placeholders`* with your own information.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.UntagResourceRequest;
import software.amazon.awssdk.services.s3control.model.UntagResourceResponse;

public class UntagResourceExample {
    public static void untagResourceExample() {
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        UntagResourceRequest untagResourceRequest = UntagResourceRequest.builder()
                .resourceArn("arn:aws::s3express:us-west-2:111122223333:bucket/my-
directory-bucket--usw2-az1--x-s3")
                .accountId("111122223333")
                .tagKeys("myTagKey")
                .build();

        UntagResourceResponse response =
 s3Control.untagResource(untagResourceRequest);
        System.out.println("Status code (should be 204):");
        System.out.println(response.sdkHttpResponse().statusCode());
    }
}
```

Using the REST API

For information about the Amazon S3 REST API support for deleting tags from a directory bucket, see the following section in the *Amazon Simple Storage Service API Reference*:

- UnTagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to delete tags from a directory bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws s3control untag-resource \
--account-id 111122223333 \
--resource-arn arn:aws::s3express:us-east-1:444455556666:bucket/prefix--use1-az4--x-s3
 \
--tag-keys "tagkey1" "tagkey2"
```

Response:

```
{
  "ResponseMetadata": {
    "RequestId": "EXAMPLE123456789",
    "HTTPStatusCode": 204,
    "HTTPHeaders": {
        "date": "Wed, 19 Jun 2025 10:30:00 GMT",
        "content-length": "0"
    },
    "RetryAttempts": 0
  }
}
```

Resilience testing in S3 Express One Zone

Amazon S3 Express One Zone storage class supports resilience testing with AWS Fault Injection Service (AWS FIS), a fully managed service for performing fault injection experiments on your AWS

workloads. With AWS FIS, you can simulate connectivity disruptions to your directory buckets, causing Zonal (object level, or data plane) endpoint API operations to timeout as would occur during an Availability Zone disruption.

These experiments can help you:

- Verify your monitoring systems can detect S3 Express One Zone access issues
- Test and strengthen recovery processes
- Validate that application failover mechanisms work as expected
- Ensure your application recovery time meets your organization's service level objectives (SLOs) and service level agreements (SLAs)

By testing your application's response to simulated disruptions, you can better prepare for the unlikely event of an actual Availability Zone outage that affects access to your data in S3 Express One Zone.

How it works

The test uses the `aws:network:disrupt-connectivity` action with scope set to `S3 Express`. This action disrupts network connectivity to S3 Express One Zone endpoints, causing requests to directory buckets to timeout.

You can target subnets where your applications are running or Gateway VPC endpoints used to access S3 Express One Zone. For more information, see AZ Availability: Power Interruption in the *AWS Fault Injection Service User Guide*.

To test the resilience of applications that store data in S3 Express One Zone, see Simulate a connectivity event in the *AWS Fault Injection Service User Guide*.

Considerations and limitations

Keep in mind the following considerations and limitations for disrupting connectivity in Amazon S3 Express One Zone storage class:

Considerations

- **IAM Permissions:** To use AWS FIS with S3 Express One Zone, you must configure an IAM role with appropriate permissions. For more information on AWS FIS roles, see Create an IAM role for

AWS FIS in the *AWS Fault Injection Service User Guide*. We recommend scoping these permissions to only the necessary resources.

- **Target Resolution:** Targets are resolved at the beginning of the experiment. If a target subnet or gateway VPC endpoint is deleted during the experiment, the experiment will fail.
- **Shared Resources Impact:** If multiple applications share the same subnets or gateway VPC endpoints, all traffic to S3 Express One Zone from these applications will be affected during the experiment.
- **Rollback Behavior:** When the AWS FIS action ends, connectivity is automatically restored and requests will resume the expected operation without manual intervention.
- **Stop Conditions:** Configure appropriate CloudWatch alarms as stop conditions to automatically terminate experiments if unexpected impacts occur.

Limitations

- **Target Selection:** You can't target specific S3 directory buckets. The action affects all directory buckets accessed through the targeted networking components.
- **Maximum Targets:** There is a maximum number of subnets you can target per AWS FIS action. For more information, see Service quotas for AWS Fault Injection Service in the *AWS FIS User Guide*.
- **Access Methods:** The AWS FIS action only affects requests made through the internet or gateway virtual private cloud (VPC) endpoints. Requests from interface VPC endpoints (AWS PrivateLink) aren't supported.
- **Regional Availability:** This feature is available only in AWS Regions where S3 Express One Zone is supported.

Working with Amazon S3 Tables and table buckets

Amazon S3 Tables provide S3 storage that's optimized for analytics workloads, with features designed to continuously improve query performance and reduce storage costs for tables. S3 Tables are purpose-built for storing tabular data, such as daily purchase transactions, streaming sensor data, or ad impressions. Tabular data represents data in columns and rows, like in a database table.

The data in S3 Tables is stored in a new bucket type: a *table bucket*, which stores tables as subresources. Table buckets support storing tables in the Apache Iceberg format. Using standard SQL statements, you can query your tables with query engines that support Iceberg, such as Amazon Athena, Amazon Redshift, and Apache Spark.

Topics

- Features of S3 Tables
- Related services
- Tutorial: Getting started with S3 Tables
- Table buckets
- S3 Tables maintenance
- Cost optimization for tables with Intelligent-Tiering
- Table namespaces
- Tables in S3 table buckets
- Accessing table data
- Working with Apache Iceberg V3
- Replicating S3 tables
- S3 Tables AWS Regions, endpoints, and service quotas
- Making requests to S3 Tables over IPv6
- Security for S3 Tables
- Logging and monitoring for S3 Tables

Features of S3 Tables

Purpose-built storage for tables

S3 table buckets are specifically designed for tables. Table buckets provide higher transactions per second (TPS) and better query throughput compared to self-managed tables in S3 general purpose buckets. Table buckets deliver the same durability, availability, and scalability as other Amazon S3 bucket types.

Built-in support for Apache Iceberg

Tables in your table buckets are stored in Apache Iceberg format. You can query these tables using standard SQL in query engines that support Iceberg. Iceberg has a variety of features to optimize query performance, including schema evolution and partition evolution.

With Iceberg, you can change how your data is organized so that it can evolve over time without requiring you to rewrite your queries or rebuild your data structures. Iceberg is designed to help ensure data consistency and reliability through its support for transactions. To help you correct issues or perform time travel queries, you can track how data changes over time and roll back to historical versions.

Your table buckets support the Apache Iceberg Version 3 (V3) table format, including deletion vectors, row lineage, column default values, and the variant, geometry, geography, unknown, and nanosecond-precision timestamp data types. For more information, see Working with Apache Iceberg V3.

Automated table optimization

To optimize your tables for querying, S3 continuously performs automatic maintenance operations, such as compaction, snapshot management, and unreferenced file removal. These operations increase table performance by compacting smaller objects into fewer, larger files. Maintenance operations also reduce your storage costs by cleaning up unused objects. This automated maintenance streamlines the operation of data lakes at scale by reducing the need for manual table maintenance. For each table and table bucket, you can customize maintenance configurations.

Access management and security

You can manage access for both table buckets and individual tables with AWS Identity and Access Management (IAM) and Service Control Policies in AWS Organizations. S3 Tables uses a different service namespace than Amazon S3: the *s3tables* namespace. Therefore, you can

design policies specifically for the S3 Tables service and its resources. You can design policies to grant access to individual tables, all tables within a table namespace, or entire table buckets. All Amazon S3 Block Public Access settings are always enabled for table buckets and cannot be disabled.

Integration with AWS analytics services

You can automatically integrate your Amazon S3 table buckets with AWS Glue Data Catalog through the S3 console. This integration allows AWS analytics services to automatically discover and access your table data. After the integration, you can work with your tables using analytics services such as Amazon Athena, Amazon Redshift, Quick, and more. For more information about how the integration works, see Integrating Amazon S3 Tables with AWS analytics services.

Related services

You can use the following AWS services with S3 Tables to support your specific analytics applications.

- **Amazon Athena** – Athena is an interactive query service that you can use to analyze data directly in Amazon S3 by using standard SQL. You can also use Athena to interactively run data analytics by using Apache Spark without having to plan for, configure, or manage resources. When you run Apache Spark applications on Athena, you submit Spark code for processing and receive the results directly.
- **AWS Glue** – AWS Glue is a serverless data-integration service that allows you to discover, prepare, move, and integrate data from multiple sources. You can use AWS Glue for analytics, machine learning (ML), and application development. AWS Glue also includes additional productivity and data-operations tooling for authoring, running jobs, and implementing business workflows.
- **Amazon SageMaker Unified Studio** – SageMaker Unified Studio delivers an integrated experience for analytics and AI with unified access to all your data. Collaborate and build in SageMaker Unified Studio using familiar AWS tools for SQL analytics, data processing, model development, and generative AI, accelerated by Amazon Q Developer.
- **Amazon EMR** – Amazon EMR is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data.

- **Amazon Redshift** – Amazon Redshift is a petabyte-scale data warehouse service in the cloud. You can use Amazon Redshift Serverless to access and analyze data without all of the configurations of a provisioned data warehouse. Resources are automatically provisioned and data warehouse capacity is intelligently scaled to deliver fast performance for even the most demanding and unpredictable workloads. You don't incur charges when the data warehouse is idle, so you only pay for what you use. You can load data and start querying right away in the Amazon Redshift query editor v2 or in your favorite business intelligence (BI) tool.
- **Quick** – Quick is a business analytics service to build visualizations, perform ad hoc analysis, and quickly get business insights from your data. Quick seamlessly discovers AWS data sources and delivers fast and responsive query performance by using the Quick Super-fast, Parallel, In-Memory, Calculation Engine (SPICE).
- **AWS Lake Formation** – Lake Formation is a managed service that streamlines the process to set up, secure, and manage your data lakes. Lake Formation helps you discover your data sources and then catalog, cleanse, and transform the data. With Lake Formation, you can manage fine-grained access control for your data lake data on Amazon S3 and its metadata in AWS Glue Data Catalog.

Tutorial: Getting started with S3 Tables

In this tutorial, you create a table bucket and integrate table buckets in your Region with AWS analytics services. Next, you will use the AWS CLI or console to create your first namespace and table in your table bucket. Then, you can begin querying your table with Athena.

> **Tip**
>
> If you're migrating tabular data from general purpose buckets to table buckets, Amazon S3 has a managed migration feature to assist you. This feature migrates AWS Glue-registered Apache Hive tables in Amazon S3 to Apache Iceberg tables in S3 Tables. For more information, see Migrating tables to S3 Tables.
> For information about migrating AWS Glue-registered Apache Iceberg tables, see Guidance for Migrating Tabular Data from Amazon S3 to S3 Tables in the AWS Solutions Library.

Topics

- Step 1: Create a table bucket and integrate it with AWS analytics services
- Step 2: Create a table namespace and a table

- Step 3: Query data with SQL in Athena

Step 1: Create a table bucket and integrate it with AWS analytics services

In this step, you use the Amazon S3 console to create your first table bucket. For other ways to create a table bucket, see Creating a table bucket.

> **Note**
>
> By default, the Amazon S3 console automatically integrates your table buckets with AWS Glue Data Catalog, which allows AWS analytics services to automatically discover and access your S3 Tables data. If you create your first table bucket programmatically by using the AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API, you must manually complete the AWS analytics services integration. For more information, see Integrating Amazon S3 Tables with AWS analytics services.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create the table bucket.
3. In the left navigation pane, choose **Table buckets**.
4. Choose **Create table bucket**.
5. Under **General configuration**, enter a name for your table bucket.

   The table bucket name must:

   - Be unique within for your AWS account in the current Region.
   - Be between 3 and 63 characters long.
   - Consist only of lowercase letters, numbers, and hyphens (-).
   - Begin and end with a letter or number.

After you create the table bucket, you can't change its name. The AWS account that creates the table bucket owns it. For more information about naming table buckets, see Table bucket naming rules.

6. In the **Integration with AWS analytics services** section, make sure that the **Enable integration** checkbox is selected.

   If **Enable integration** is selected when you create your first table bucket by using the console, Amazon S3 attempts to integrate your table bucket with AWS analytics services. This integration allows you to use AWS analytics services to access all tables in the current Region. For more information, see Integrating Amazon S3 Tables with AWS analytics services.

7. Choose **Create bucket**.

Step 2: Create a table namespace and a table

For this step, you create a namespace in your table bucket, and then create a new table under that namespace. You can create a table namespace and a table by using either the console or the AWS CLI.

> **Important**
>
> When creating tables, make sure that you use all lowercase letters in your table names and table definitions. For example, make sure that your column names are all lowercase. If your table name or table definition contains capital letters, the table isn't supported by AWS Lake Formation or the AWS Glue Data Catalog. In this case, your table won't be visible to AWS analytics services such as Amazon Athena, even if your table buckets are integrated with AWS analytics services.
>
> If your table definition contains capital letters, you receive the following error message when running a SELECT query in Athena: "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names."

Using the S3 console and Amazon Athena

The following procedure uses the Amazon S3 console to create a namespace and a table with Amazon Athena.

To create a table namespace and a table

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. On the **Table buckets** page, choose the table bucket that you want to create a table in.
4. On the table bucket details page, choose **Create table with Athena**.
5. In the **Create table with Athena** dialog box, choose **Create a namespace**, and then enter a name in the **Namespace name** field. Namespace names must be 1 to 255 characters and unique within the table bucket. Valid characters are a–z, 0–9, and underscores (_). Underscores aren't allowed at the start of namespace names.
6. Choose **Create namespace**.
7. Choose **Create table with Athena**.
8. The Amazon Athena console opens and the Athena query editor appears. The query editor is populated with a sample query that you can use to create a table. Modify the query to specify the table name and columns that you want your table to have.
9. When you're finished modifying the query, choose **Run** to create your table.

If your table creation was successful, the name of your new table appears in the list of tables in Athena. When you navigate back to the Amazon S3 console, your new table appears in the **Tables** list on the details page for your table bucket after you refresh the list.

Using the AWS CLI

To use the following AWS CLI example commands to create a namespace in your table bucket, and then create a new table with a schema under that namespace, replace the *`user input placeholder`* values with your own.

Prerequisites

- Attach the `AmazonS3TablesFullAccess` policy to your IAM identity.
- Install AWS CLI version 2.23.10 or higher. For more information, see Installing or updating the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

1. Create a new namespace in your table bucket by running the following command:

```
aws s3tables create-namespace \
--table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket \
--namespace my_namespace
```

- Confirm that your namespace was created successfully by running the following command:

  ```
  aws s3tables list-namespaces \
  --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket
  ```

2. Create a new table with a table schema by running the following command:

   ```
   aws s3tables create-table --cli-input-json file://mytabledefinition.json
   ```

   For the `mytabledefinition.json` file, use the following example table definition:

   ```
   {
       "tableBucketARN": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket",
       "namespace": "my_namespace",
       "name": "my_table",
       "format": "ICEBERG",
       "metadata": {
           "iceberg": {
               "schema": {
                   "fields": [
                        {"name": "id", "type": "int","required": true},
                        {"name": "name", "type": "string"},
                        {"name": "value", "type": "int"}
                   ]
               }
           }
       }
   }
   ```

Step 3: Query data with SQL in Athena

You can query your table with SQL in Athena. Athena supports Data Definition Language (DDL), Data Manipulation Language (DML), and Data Query Language (DQL) queries for S3 Tables.

You can access the Athena query either from the Amazon S3 console or through the Amazon Athena console.

Using the S3 console and Amazon Athena

The following procedure uses the Amazon S3 console to access the Athena query editor so that you can query a table with Amazon Athena.

To query a table

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. On the **Table buckets** page, choose the table bucket that contains the table that you want to query.
4. On the table bucket details page, choose the option button next to the name of the table that you want to query.
5. Choose **Query table with Athena**.
6. The Amazon Athena console opens and the Athena query editor appears with a sample `SELECT` query loaded for you. Modify this query as needed for your use case.
7. To run the query, choose **Run**.

Using the Amazon Athena console

To query a table

1. Open the Athena console at https://console.aws.amazon.com/athena/.
2. Query your table. The following is a sample query that you can modify. Make sure to replace the *`user input placeholders`* with your own information.

```
SELECT * FROM "s3tablescatalog/amzn-s3-demo-table-bucket"."my_namespace"."my_table"
 LIMIT 10
```

3. To run the query, choose **Run**.

Table buckets

Amazon S3 table buckets are an S3 bucket type that you can use to create and store tables as S3 resources. Table buckets are used to store tabular data and metadata as objects for use in analytics workloads. S3 performs maintenance in your table buckets automatically to help reduce your table storage costs. For more information, see S3 Tables maintenance.

To interact with the tables stored inside your table buckets, you can integrate your table buckets with analytics applications that support Apache Iceberg. Table buckets integrate with AWS analytics services through the AWS Glue Data Catalog. For more information, see Integrating Amazon S3 Tables with AWS analytics services. You can also interact with your tables using open-source query engines using the Amazon S3 Tables Catalog for Apache Iceberg. For more information, see Accessing tables using the Amazon S3 Tables Iceberg REST endpoint.

Each table bucket has a unique Amazon Resource Name (ARN) and resource policy attached to it. Table bucket ARNs follow this format:

```
arn:aws:s3tables:Region:OwnerAccountID:bucket/bucket-name
```

All table buckets and tables are private and can't be made public. These resources can only be accessed by users who are explicitly granted access. To grant access, you can use IAM resource-based policies for table buckets and tables, and IAM identity-based policies for users and roles.

By default, you can create up to 100 table buckets per AWS Region in an AWS account. To request a quota increase for table buckets or tables, contact Support.

Types of table buckets

Amazon S3 supports the following types of table buckets:

Customer-managed table buckets

Customer-managed table buckets are resources for storing Amazon S3 Tables created and managed by customers. You create these buckets explicitly, choose their names, and maintain full control over the tables and namespaces within them. For customer-managed table buckets, you can create, delete, set custom default encryption, or configure maintenance options as needed.

AWS managed table buckets

AWS managed table buckets are AWS managed resources that automatically store tables created by AWS services, such as the live inventory and journal tables created by S3 Metadata. These buckets provide a centralized location for all system-generated tables. These buckets follow a standard naming convention, use a standard namespace for all tables, and have preset maintenance and default encryption configurations which S3 modifies on your behalf. You have read-only access to query the data, while AWS handles all table creation, updates, and maintenance operations. For more information, see Working with AWS managed table buckets.

There are several types of Amazon S3 buckets. Before creating a bucket, make sure that you choose the bucket type that best fits your application and performance requirements. For more information about the various bucket types and the appropriate use cases for each, see Buckets.

Topics

- Amazon S3 table bucket, table, and namespace naming rules
- Creating a table bucket
- Deleting a table bucket
- Viewing details about an Amazon S3 table bucket
- Managing table bucket policies
- Working with AWS managed table buckets
- Using tags with S3 table buckets

Amazon S3 table bucket, table, and namespace naming rules

When you create a table bucket, you choose a bucket name and AWS Region, the name must be unique for your account in the chosen Region. After you create a table bucket, you can't change the bucket name or Region. Table bucket names must follow specific naming rules. For more information about naming rules for table buckets and the tables and namespaces within them, see the following topic.

Topics

- Table bucket naming rules
- Naming rules for tables and namespaces

Table bucket naming rules

When you create Amazon S3 table buckets, you specify a table bucket name. Like other bucket types, table buckets can't be renamed. Unlike other bucket types, table buckets aren't in a global namespace, so each bucket name in your account needs to be unique only within your current AWS Region.

For general purpose bucket naming rules, see General purpose bucket naming rules. For directory bucket naming rules, see Directory bucket naming rules.

The following naming rules apply for table buckets.

- Bucket names must be between 3 and 63 characters long.
- Bucket names can consist only of lowercase letters, numbers, and hyphens (-).
- Bucket names must begin and end with a letter or number.
- Bucket names must not contain any underscores (_) or periods (.).
- Bucket names must not start with any of the following reserved prefixes:
  - `xn--`
  - `sthree-`
  - `amzn-s3-demo-`
  - `aws`
- Bucket names must not end with any of the following reserved suffixes:
  - `-s3alias`
  - `--ol-s3`
  - `--x-s3`
  - `--table-s3`

Naming rules for tables and namespaces

The following naming rules apply to tables and namespaces within table buckets:

- Names must be between 1 and 255 characters long.
- Names can consist only of lowercase letters, numbers, and underscores (_).
- Names must begin with a letter or number.
- Names must not contain hyphens (-) or periods (.).

- A table name must be unique within a namespace.
- A namespace must be unique within a table bucket.
- Namespace names must not start with the reserved prefix `aws`.

> **Important**
>
> When creating tables, make sure that you use all lowercase letters in your table names and table definitions. For example, make sure that your column names are all lowercase. If your table name or table definition contains capital letters, the table isn't supported by AWS Lake Formation or the AWS Glue Data Catalog. In this case, your table won't be visible to AWS analytics services such as Amazon Athena, even if your table buckets are integrated with AWS analytics services.
>
> If your table definition contains capital letters, you receive the following error message when running a `SELECT` query in Athena: "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names."

Creating a table bucket

Amazon S3 table buckets are an S3 bucket type that you can use to create and store tables as S3 resources. To start using S3 Tables, you create a table bucket where you store and manage tables. When you create a table bucket, you choose a bucket name and AWS Region. The table bucket name must be unique for your account in the chosen Region. After you create a table bucket, you can't change the bucket name or Region. For information about naming table buckets, see Amazon S3 table bucket, table, and namespace naming rules.

Table buckets have the following Amazon Resource Name (ARN) format:

```
arn:aws:s3tables:region:owner-account-id:bucket/bucket-name
```

By default, you can create up to 100 table buckets per Region in an AWS account. To request a quota increase for table buckets or tables, contact Support.

When you create a table bucket you can specify the encryption type for that will be used to encrypt the tables you create in that bucket. For more information about bucket encryption options, see the section called “Encryption”.

Prerequisites for creating table buckets

To create a table bucket, you must first do the following:

- Make sure that you have AWS Identity and Access Management (IAM) permissions for `s3tables:CreateTableBucket`.

> **Note**
>
> If you choose SSE-KMS as the default encryption type, you must have permissions for `s3tables:PutTableBucketEncryption`, and have `DescribeKey` permission on the chosen AWS KMS key. Additionally the AWS KMS key you use needs to grant S3 Tables permission to perform automatic table maintenance. For more information, see Permission requirements for S3 Tables SSE-KMS encryption.

To create a table bucket, you can use the Amazon S3 console, Amazon S3 REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.
3. In the left navigation pane, choose **Table buckets**.
4. Choose **Create table bucket** to open the **Create table bucket** page.
5. Under **Properties**, enter a name for your table bucket.

   The table bucket name must:

   - Be unique within for your account in the current Region.
   - Be between 3 and 63 characters long
   - Consist only of lowercase letters, numbers, and hyphens (-).
   - Begin and end with a letter or number.

After you create the bucket, you can't change its name. The AWS account that creates the bucket owns it. For information about naming table buckets, see Amazon S3 table bucket, table, and namespace naming rules.

6. If you want to integrate your table buckets with AWS analytics services, make sure **Enable integration** is selected under **Integration with AWS analytics services**.

> **Note**
>
> When you create your first table bucket by using the console with the **Enable integration** option selected, Amazon S3 attempts to automatically integrate your table bucket with AWS analytics services. This integration allows you to use AWS analytics services to query all tables in the current Region. For more information, see Integrating Amazon S3 Tables with AWS analytics services.

7. To configure default encryption, under **Encryption type**, choose one of the following:

   - **Server-side encryption with Amazon S3 managed key (SSE-S3)**
   - **Server-side encryption with AWS Key Management Service key (SSE-KMS)**

   For more information about encryption options for table data, see Protecting S3 table data with encryption.

8. Choose **Create bucket**.

Using the AWS CLI

This example shows how to create a table bucket by using the AWS CLI. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3tables create-table-bucket \
    --region us-east-2 \
    --name amzn-s3-demo-bucket1
```

By default S3 table buckets use SSE-S3 as their default encryption setting, however, you can use the optional `--encryption-configuration` parameter to specify a different encryption type.

The following examples shows how to create a bucket that uses SSE-KMS encryption. For more information on encryption settings for table buckets, see Protecting S3 table data with encryption.

```
aws s3tables create-table-bucket \
    --region us-east-2 \
    --name amzn-s3-demo-bucket1
    --encryption-configuration '{
                    "sseAlgorithm": "aws:kms",
                    "kmsKeyArn":
 "arn:aws:kms:Region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab" }'
```

Deleting a table bucket

You can use the Amazon S3 APIs, AWS Command Line Interface, or AWS SDKs to delete a table bucket. Before you delete a table bucket, you must first delete all namespaces and tables within the bucket.

Important

When you delete a table bucket, you need to know the following:

- Bucket deletion is permanent and can't be undone.
- All data and configurations associated with the bucket are permanently lost.

Using the AWS CLI

This example shows how to delete a table bucket by using the AWS CLI. To use this example, replace the *user input placeholders* with your own information.

```
aws s3tables delete-table-bucket \
    --region us-east-2 \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket1
```

Viewing details about an Amazon S3 table bucket

You can view the general details of an Amazon S3 table bucket, such as bucket owner and type, in the console or programmatically. You can view default encryption settings, and maintenance settings programmatically by using the S3 Tables REST API, AWS CLI or AWS SDKs.

Viewing table bucket details

Using the AWS CLI

This example shows how to get details about a table bucket by using the AWS CLI. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3tables get-table-bucket --table-bucket-arn arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-table-bucket
```

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. Select your table bucket.
4. Select the **Properties** tab.

Viewing table bucket encryption settings

For more information about table bucket encryption, see Using server-side encryption with AWS KMS keys (SSE-KMS) in table buckets.

Using the AWS CLI

This example shows how to get details about encryption settings for a table bucket by using the AWS CLI. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3tables get-table-bucket-encryption --table-bucket-arn arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-table-bucket
```

Viewing table bucket maintenance configurations

For information about maintenance settings, see Maintenance for table buckets

Using the AWS CLI

This example shows how to get details about maintenance configuration settings for a table bucket by using the AWS CLI. To use this example, replace the *user input placeholders* with your own information.

```
aws s3tables get-table-bucket-maintenance-configuration --table-bucket-arn
 arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket
```

Managing table bucket policies

You can add, delete, update, and view bucket policies for Amazon S3 table buckets by using the Amazon S3 REST API, AWS SDKs, and the AWS Command Line Interface (AWS CLI). For more information, see the following topics.

For more information, see the following topics. For more information about supported AWS Identity and Access Management (IAM) actions and condition keys for Amazon S3 Tables, see Access management for S3 Tables. For example bucket policies for table buckets, see Resource-based policies for S3 Tables.

> **Note**
>
> The table bucket policy provides access to the tables stored in the bucket. Table bucket policies don't apply to tables owned by other accounts.

Adding a table bucket policy

To add a bucket policy to a table bucket, use the following AWS CLI example.

Using the AWS CLI

This example shows how to create a table bucket policy by using the AWS CLI. To use the command, replace the *user input placeholders* with your own information.

```
aws s3tables put-table-bucket-policy \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket1  \
    --resource-policy your-policy-JSON
```

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Amazon S3**.
3. Choose **Table buckets** and select the table bucket name that you want to add a policy to.
4. Choose the **Permissions** tab.
5. Under **Table bucket policy**, Choose **Edit**.
6. In the policy editor, enter your policy JSON.
7. (Optional) Choose **Policy examples** to see sample policies that you can adapt to your needs.
8. After entering your policy, choose **Save changes**.

Viewing a table bucket policy

To view the bucket policy that's attached to a table bucket, use the following AWS CLI example.

Using the AWS CLI

This example shows how to view the policy that's attached to a table bucket by using the AWS CLI. To use the command, replace the *`user input placeholders`* with your own information.

```
aws s3tables get-table-bucket-policy --table-bucket-arn arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-bucket1
```

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Amazon S3**.
3. Choose **Table buckets** and select the table bucket name that you want to view the policy for.
4. Choose the **Permissions** tab.

Deleting a table bucket policy

To delete a bucket policy that's attached to a table bucket, use the following AWS CLI example.

Using the AWS CLI

This example shows how to delete a table bucket policy by using the AWS CLI. To use the command, replace the *user input placeholders* with your own information.

```
aws s3tables delete-table-bucket-policy --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-bucket1
```

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Amazon S3**.
3. Choose **Table buckets** and select the table bucket name that you want to delete a policy for.
4. Choose the **Permissions** tab.
5. Under **Table bucket policy**, choose **Delete**.

Working with AWS managed table buckets

AWS managed table buckets are specialized Amazon S3 table buckets designed to store AWS managed tables, such as S3 Metadata journal and live inventory tables. Unlike customer-managed table buckets that you create and manage directly, AWS managed table buckets are automatically provisioned by AWS when you configure features that require AWS managed tables. When managed tables are created, they belong to a predefined namespace that's based on the name of the source bucket. This predefined namespace can't be modified.

Each AWS account has one AWS managed table bucket per Region, following the naming convention `aws-s3`. This bucket serves as a centralized location for all managed tables associated with your account's resources in that Region.

The following table compares AWS managed table buckets with customer-managed table buckets.

Feature	AWS managed table buckets	Customer-managed table buckets
Creation	Automatically created by AWS services	You create these manually
Naming	Use a standard naming convention (`aws-s3`)	You define your own names

Feature	AWS managed table buckets	Customer-managed table buckets
Table creation	Only AWS services can create tables	You can create tables
Namespace control	You can't create or delete namespaces (all tables belong to a fixed namespace)	You can create and delete namespaces
Access	Read-only access	Full access
Encryption	You can change the default encryption (SSE-S3) settings only if you encrypted the initial table with a customer managed AWS Key Management Service (AWS KMS) key	You can set bucket-level default encryption and modify it anytime
Maintenance	Managed by AWS services	You can customize automated maintenance at the bucket level

Permissions to work with AWS managed table buckets and to query tables

To work with AWS managed table buckets, you need permissions to create AWS managed table buckets and tables and to specify encryption settings for AWS managed tables. You also need permissions to query the tables in your AWS managed table buckets.

The following example policy allows you to create an AWS managed table bucket through an S3 Metadata configuration:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"PermissionsToWorkWithMetadataTables",
         "Effect":"Allow",
         "Action":[
             "s3:CreateBucketMetadataTableConfiguration",
             "s3tables:CreateTableBucket",
             "s3tables:CreateNamespace",
             "s3tables:CreateTable",
             "s3tables:GetTable",
```

```
                "s3tables:PutTablePolicy",
                "s3tables:PutTableEncryption",
                "kms:DescribeKey"
            ],
            "Resource":[
               "arn:aws:s3:::bucket/amzn-s3-demo-source-bucket",
               "arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3",
               "arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3/table/*",
               "arn:aws:kms:us-east-1:111122223333:key/01234567-89ab-
cdef-0123-456789abcdef"
            ]
         }
      ]
}
```

The following example policy allows you to query tables in AWS managed table buckets:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"PermissionsToQueryMetadataTables",
         "Effect":"Allow",
         "Action":[
             "s3tables:GetTable",
             "s3tables:GetTableData",
             "s3tables:GetTableMetadataLocation",
             "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3",
            "arn:aws:s3tables:us-east-1:111122223333:bucket/aws-s3/table/*",
            "arn:aws:kms:us-east-1:111122223333:key/01234567-89ab-
cdef-0123-456789abcdef"
         ]
      }
   ]
}
```

Querying tables in AWS managed table buckets

You can query AWS managed tables in AWS managed table buckets using access methods and engines supported by S3 Tables. The following are some example queries.

Using standard SQL

The following example shows how to query AWS managed tables using standard SQL syntax:

```
SELECT *
FROM "s3tablescatalog/aws-s3"."b_amzn-s3-demo-source-bucket"."inventory"
LIMIT 10;
```

The following example shows how to join AWS managed tables with your own tables:

```
SELECT *
FROM "s3tablescatalog/aws-s3"."b_amzn-s3-demo-source-bucket"."inventory" a
JOIN "s3tablescatalog/amzn-s3-demo-table-bucket"."my_namespace"."my_table" b
ON a.key = b.key
LIMIT 10;
```

Using Spark

The following example shows how to query your table with Spark:

```
spark.sql("""
    SELECT *
    FROM ice_catalog.inventory a
    JOIN ice_catalog.my_table b
    ON a.key = b.key
""").show(10, true)
```

The following example shows how to joining your AWS managed table with another table:

```
SELECT *
FROM inventory a
JOIN my_table b
ON a.key = b.key
LIMIT 10;
```

Encryption for AWS managed table buckets

By default, AWS managed table buckets are encrypted with server-side encryption using Amazon S3 managed keys (SSE-S3). After your AWS managed table bucket is created, you can use

PutTableBucketEncryption to set the bucket's default encryption setting to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS).

During creation of your AWS managed tables, you can choose to encrypt them with SSE-KMS. If you choose to use SSE-KMS, you must provide a customer managed KMS key in the same Region as your AWS managed table bucket. You can set the encryption type for your AWS managed tables only during table creation. After an AWS managed table is created, you can't change its encryption setting.

If you want the AWS managed table bucket and the tables stored in it to use the same KMS key, make sure to use the same KMS key that you used to encrypt your tables to encrypt your table bucket after it's been created. After you've changed the default encryption settings for your table bucket to use SSE-KMS, those encryption settings are used for any future tables that are created in the bucket.

Using tags with S3 table buckets

An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 table buckets. You can tag S3 table buckets when you create them or manage tags on existing table buckets. For general information about tags, see Tagging for cost allocation or attribute-based access control (ABAC).

> **Note**
>
> There is no additional charge for using tags on table buckets beyond the standard S3 API request rates. For more information, see Amazon S3 pricing.

Common ways to use tags with table buckets

Use tags on your S3 table buckets for:

Attribute-based access control (ABAC) – Scale access permissions and grant access to S3 table buckets based on their tags. For more information, see Using tags for ABAC.

ABAC for S3 table buckets

Amazon S3 table buckets support attribute-based access control (ABAC) using tags. Use tag-based condition keys in your AWS organizations, AWS Identity and Access Management (IAM), and S3 table bucket policies. ABAC in Amazon S3 supports authorization across multiple AWS accounts.

In your IAM policies, you can control access to S3 table buckets based on the table bucket's tags by using the `s3tables:TableBucketTag/tag-key` condition key or the AWS global condition keys: `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys`.

aws:ResourceTag/key-name

Use this condition key to compare the tag key-value pair that you specify in the policy with the key-value pair attached to the resource. For example, you could require that access to a table bucket is allowed only if the table bucket has the tag key `Department` with the value `Marketing`.

This condition key applies to all table bucket actions that are performed using the Amazon S3 Console, the AWS Command Line Interface (CLI), S3 APIs, or the AWS SDKs, except for the `CreateBucket` API request.

For an example policy, see 1.1 - table bucket policy to restrict operations on the tables inside of the table bucket using tags.

For additional example policies and more information, see Controlling access to AWS resources in the *AWS Identity and Access Management User Guide*.

> **Note**
>
> For actions performed on tables, this condition key acts on the tags applied to the table and not on the tags applied to the table bucket containing the table. Use the `s3tables:TableBucketTag/tag-key` instead if you would like your ABAC policies to act on the tags of the table bucket when performaing table actions.

aws:RequestTag/key-name

Use this condition key to compare the tag key-value pair that was passed in the request with the tag pair that you specify in the policy. For example, you could check whether the request to tag a table bucket includes the tag key `Department` and that it has the value `Accounting`.

This condition key applies when tag keys are passed in a `TagResource` or `CreateTableBucket` API operation request, or when tagging or creating a table bucket with tags using the Amazon S3 Console, the AWS Command Line Interface (CLI), or the AWS SDKs.

For an example policy, see 1.2 - IAM policy to create or modify table buckets with specific tags.

For additional example policies and more information, see Controlling access during AWS requests in the *AWS Identity and Access Management User Guide*.

aws:TagKeys

Use this condition key to compare the tag keys in a request with the keys that you specify in the policy to define what tag keys are allowed for access. For example, to allow tagging during the `CreateTableBucket` action, you must create a policy that allows both the `s3tables:TagResource` and `s3tables:CreateTableBucket` actions. You can then use the `aws:TagKeys` condition key to enforce that only specific tags are used in the `CreateTableBucket` request.

This condition key applies when tag keys are passed in a `TagResource`, `UntagResource`, or `CreateTableBucket` API operations or when tagging, untagging, or creating a table bucket with tags using the Amazon S3 Console, the AWS Command Line Interface (CLI), or the AWS SDKs.

For an example policy, see 1.3 - IAM policy to control the modification of tags on existing resources maintaining tagging governance.

For additional example policies and more information, see Controlling access based on tag keys in the *AWS Identity and Access Management User Guide*.

s3tables:TableBucketTag/tag-key

Use this condition key to grant permissions to specific data in table buckets using tags. This condition key acts, on the most part, on the tags assigned to the table bucket for all S3 tables actions. Even when you create a table with tags, this condition key acts on the tags applied to the table bucket that contains that table. The exceptions are:

- When you create a table bucket with tags, this condition key acts on the tags in the request.

For an example policy, see 1.4 - Using the s3tables:TableBucketTag condition key.

Example ABAC policies for table buckets

See the following example ABAC policies for Amazon S3 table buckets.

> **Note**
>
> If you are using Lake Formation to manage access to your Amazon S3 Tables and you have an IAM or S3 Tables resource-based policy that restricts IAM users and IAM roles based on

principal tags, you must attach the same principal tags to the IAM role that Lake Formation uses to access your Amazon S3 data (for example, LakeFormationDataAccessRole) and grant this role the necessary permissions. This is required for your tag-based access control policy to work correctly with your S3 Tables analytics integration.

1.1 - table bucket policy to restrict operations on the tables inside of the table bucket using tags

In this table bucket policy, the specified IAM principals (users and roles) can perform the `GetTable` action on any table in the table bucket only if the value of the table's `project` tag matches the value of the principal's `project` tag.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGetTable",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "s3tables:GetTable",
      "Resource": "arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        }
      }
    }
  ]
}
```

1.2 - IAM policy to create or modify table buckets with specific tags

Note

If you are using AWS Lake Formation to manage access to your Amazon S3 tables, and you are using ABAC with Amazon S3 Tables, make sure that you also give the IAM role that Lake Formation assumes the required access. For more information on setting up the IAM role

for Lake Formation, see Prerequisites for integrating Amazon S3 tables catalog with the Data Catalog and Lake Formation in the *AWS Lake Formation developer guide*.

In this IAM policy, users or roles with this policy can only create S3 table buckets if they tag the table bucket with the tag key `project` and tag value `Trinity` in the table bucket creation request. They can also add or modify tags on existing S3 table buckets as long as the `TagResource` request includes the tag key-value pair `project:Trinity`. This policy does not grant read, write, or delete permissions on the table buckets or its objects.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateTableBucketWithTags",
      "Effect": "Allow",
      "Action": [
        "s3tables:CreateTableBucket",
        "s3tables:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/project": [
            "Trinity"
          ]
        }
      }
    }
  ]
}
```

1.3 - IAM policy to control the modification of tags on existing resources maintaining tagging governance

In this IAM policy, IAM principals (users or roles) can modify tags on a table bucket only if the value of the table bucket's `project` tag matches the value of the principal's `project` tag. Only the four tags `project`, `environment`, `owner`, and `cost-center` specified in the `aws:TagKeys` condition keys are permitted for these table buckets. This helps enforce tag governance, prevents unauthorized tag modifications, and keeps the tagging schema consistent across your table buckets.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceTaggingRulesOnModification",
      "Effect": "Allow",
      "Action": [
        "s3tables:TagResource",
        "s3tables:UntagResource"
      ],
      "Resource": "arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-
bucket",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "project",
            "environment",
            "owner",
            "cost-center"
          ]
        }
      }
    }
  ]
}
```

1.4 - Using the s3tables:TableBucketTag condition key

In this IAM policy, the condition statement allows access to the table bucket's data only if the table bucket has the tag key `Environment` and tag value `Production`. The `s3tables:TableBucketTag/<tag-key>` differs from the `aws:ResourceTag/<tag-key>` condition key because, in addition to controlling access to table buckets depending on their tags, it allows you to control access to tables based on the tags on their parent table bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToSpecificTables",
```

```
      "Effect": "Allow",
      "Action": "*",
      "Resource": "arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-
bucket/*",
      "Condition": {
        "StringEquals": {
          "s3tables:TableBucketTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Managing tags for table buckets

You can add or manage tags for S3 table buckets using the Amazon S3 Console, the AWS Command Line Interface (CLI), the AWS SDKs, or using the S3 APIs: TagResource, UntagResource, and ListTagsForResource. For more information, see the following:

Topics

- Creating table buckets with tags
- Adding a tag to a table bucket
- Viewing table bucket tags
- Deleting a tag from a table bucket

Creating table buckets with tags

You can tag Amazon S3 table buckets when you create them. There is no additional charge for using tags on table buckets beyond the standard S3 API request rates. For more information, see Amazon S3 pricing. For more information about tagging table buckets, see Using tags with S3 table buckets.

Permissions

To create a table bucket with tags, you must have the following permissions:

- `s3tables:CreateTableBucket`
- `s3tables:TagResource`

Troubleshooting errors

If you encounter an error when attempting to create a table bucket with tags, you can do the following:

- Verify that you have the required Permissions to create the table bucket and apply a tag to it.
- Check your IAM user policy for any attribute-based access control (ABAC) conditions. Your policy may require you to tag your table buckets with only specific tag keys and values. For more information about ABAC and example table bucket ABAC policies, see ABAC for S3 table buckets.

Steps

You can create a table bucket with tags applied by using the Amazon S3 Console, the AWS Command Line Interface (AWS CLI), the Amazon S3 Tables REST API, and the AWS SDKs.

Using the S3 console

To create a table bucket with tags using the Amazon S3 console:

1. Sign in to the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. To create a new table bucket, choose **Create table bucket**.
4. Enter a name for the table bucket. For more information, see Amazon S3 table bucket, table, and namespace naming rules.
5. On the **Create table bucket** page, there is a **Tags** section.
6. Choose **Add new Tag** to open the Tags editor and enter a tag key-value pair. The tag key is required, but the value is optional.
7. To add another tag, select **Add new Tag** again. You can enter up to 50 tag key-value pairs.
8. Specify the remaining options for your new table bucket. For more information, see Creating a table bucket.
9. Choose **Create table bucket**.

Using the REST API

For information about the Amazon S3 Tables REST API support for creating a table bucket with tags, see the following section in the *Amazon Simple Storage Service API Reference*:

- CreateTableBucket

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to create a table bucket with tags by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

When you create a table bucket you must provide configuration details. For more information, see Creating a table bucket. You must also name the table bucket with a name that follows the table bucket naming convention. For more information see Amazon S3 table bucket, table, and namespace naming rules.

Request:

```
aws --region us-west-2 \
s3tables create-table-bucket \
--tags '{"Department":"Engineering"}' \
--name amzn-s3-demo-table-bucket
```

Adding a tag to a table bucket

You can add tags to Amazon S3 table buckets and modify these tags. For more information about tagging table buckets, see Using tags with S3 table buckets.

Permissions

To add a tag to a table bucket, you must have the following permission:

- `s3tables:TagResource`

Troubleshooting errors

If you encounter an error when attempting to add a tag to a table bucket, you can do the following:

- Verify that you have the required Permissions to add a tag to a table bucket.
- If you attempted to add a tag key that starts with the AWS reserved prefix `aws:`, change the tag key and try again.

- The tag key is required. Also, make sure that the tag key and the tag value do not exceed the maximum character length and do not contain restricted characters. For more information, see Tagging for cost allocation or attribute-based access control (ABAC).

Steps

You can add tags to table buckets by using the Amazon S3 Console, the AWS Command Line Interface (AWS CLI), the Amazon S3 Tables REST API, and the AWS SDKs.

Using the S3 console

To add tags to a table bucket using the Amazon S3 console:

1. Sign in to the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. Choose the table bucket name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and choose **Add new Tag**.
6. This opens the **Add Tags** page. You can enter up to 50 tag key value pairs.
7. If you add a new tag with the same key name as an existing tag, the value of the new tag overrides the value of the existing tag.
8. You can also edit the values of existing tags on this page.
9. After you have added the tag(s), choose **Save changes**.

Using the REST API

For information about the Amazon S3 REST API support for adding tags to a table bucket, see the following section in the *Amazon Simple Storage Service API Reference*:

- TagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to add tags to a table bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws --region us-west-2 \
s3tables tag-resource \
--resource-arn arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket \
--tags '{"Department":"Engineering"}'
```

Viewing table bucket tags

You can view or list tags applied to Amazon S3 table buckets. For more information about tagging table buckets, see Using tags with S3 table buckets.

Permissions

To view tags applied to a table bucket, you must have the following permission:

- `s3tables:ListTagsForResource`

Troubleshooting errors

If you encounter an error when attempting to list or view the tags of a table bucket, you can do the following:

- Verify that you have the required Permissions to view the tags of the table bucket.

Steps

You can view tags applied to table buckets by using the Amazon S3 Console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

Using the S3 console

To view tags applied to a table bucket using the Amazon S3 console:

1. Sign in to the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. Choose the table bucket name.
4. Choose the **Properties** tab.

5. Scroll to the **Tags** section to view all of the tags applied to the table bucket.
6. The **Tags** section shows the **User-defined tags** by default. You can select the **AWS-generated tags** tab to view tags applied to your table bucket by AWS services.

Using the REST API

For information about the Amazon S3 REST API support for viewing the tags applied to a table bucket, see the following section in the *Amazon Simple Storage Service API Reference*:

- ListTagsforResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to view tags applied to a table bucket. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws --region us-west-2 \
s3tables list-tags-for-resource \
--resource-arn arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket/table/example_table
```

Response - tags present:

```
{
    "tags": {
        "project": "Trinity",
        "code": "123456"
    }
}
```

Response - no tags present:

```
{
  "Tags": []
}
```

Deleting a tag from a table bucket

You can remove tags from Amazon S3 table buckets. For more information about tagging table buckets, see Using tags with S3 table buckets.

> **Note**
>
> If you delete a tag and later learn that it was being used to track costs or for access control, you can add the tag back to the table bucket.

Permissions

To delete a tag from a table bucket, you must have the following permission:

- `s3tables:UntagResource`

Troubleshooting errors

If you encounter an error when attempting to delete a tag from a table bucket, you can do the following:

- Verify that you have the required Permissions to delete a tag from a table bucket.

Steps

You can delete tags from table buckets by using the Amazon S3 Console, the AWS Command Line Interface (AWS CLI), the Amazon S3 Tables REST API, and the AWS SDKs.

Using the S3 console

To delete tags from a table bucket using the Amazon S3 console:

1. Sign in to the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. Choose the table bucket name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and select the checkbox next to the tag or tags that you would like to delete.

6. Choose **Delete**.
7. The **Delete user-defined tags** pop-up appears and asks you to confirm the deletion of the tag or tags you selected.
8. Choose **Delete** to confirm.

Using the REST API

For information about the Amazon S3 REST API support for deleting tags from a table bucket, see the following section in the *Amazon Simple Storage Service API Reference*:

- UnTagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to delete tags from a table bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws --region us-west-2 \
s3tables untag-resource \
--resource-arn arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket \
--tags-keys '["Department"]'
```

S3 Tables maintenance

Amazon S3 automatically performs maintenance to enhance the performance of your tables in S3 table buckets. Maintenance is performed at the table bucket and individual table level and includes the following:

Table bucket-level maintenance:

- **Unreferenced file removal** – Cleans up orphaned files to optimize storage usage and costs.

Table-level maintenance:

- **File compaction** – Consolidates small files to improve query performance and reduce storage costs.

- **Snapshot management** – Controls table version history and prevents excessive metadata growth.

These options are enabled by default. You can edit or disable these operations through maintenance configuration files.

In addition to these options, you can also enable and configure record expiration settings for tables. With this option, Amazon S3 automatically removes records from a table when the records expire.

Topics

- S3 Tables maintenance job status
- Maintenance for table buckets
- Maintenance for tables
- Record expiration for tables
- Considerations and limitations for maintenance jobs

S3 Tables maintenance job status

S3 Tables maintenance jobs run periodically for your S3 tables or table buckets. You can query the status of these jobs with the `GetTableMaintenanceJobStatus` API.

To get the status of your maintenance jobs by using the AWS CLI

The following example will get the statuses of maintenance jobs using the `GetTableMaintenanceJobStatus` API.

```
aws s3tables get-table-maintenance-job-status \
   --table-bucket-arn="arn:aws:s3tables:arn:aws::111122223333:bucket/amzn-s3-demo-bucket1" \
   --namespace="mynamespace" \
   --name="testtable"
```

For more information, see get-table-maintenance-job-status in the *AWS CLI Command Reference*.

S3 Tables maintenance jobs can transition between four possible statuses:

- `Successful`
- `Failed`
- `Disabled`
- `Not_Yet_Run`

Jobs with a failed status will include a failure message. The following list describes possible failure messages.

- Encountered Iceberg validation exception when trying to read table. Ensure that your table is readable, adheres to the Iceberg specification, and contains only S3 paths that begin with your S3 Table alias.
- Iceberg Snapshot management does not currently support user defined tags or references.
- Iceberg table maintenance configuration is incompatible with 'history.expire.max-snapshot-age-ms' and 'history.expire.min-snapshots-to-keep' table properties.
- Iceberg snapshot management and unreferenced file removal is not supported when the 'gc.enabled' table property is false. Ensure that this property is unset or explicitly set to true.
- Failed to commit because of out of date metadata. Maintenance will be retried at the next available opportunity.
- Insufficient access to perform table maintenance. Ensure that the key used to encrypt the table is active, exists, and has a resource policy granting access to the S3 service principal `maintenance.s3tables.amazonaws.com`.

> **Note**
>
> For more information on AWS KMS permissions for S3 Tables, see Permission requirements for S3 Tables SSE-KMS encryption.

- Internal error

Maintenance for table buckets

Amazon S3 offers maintenance operations to enhance the management and performance of your table buckets. The following option is enabled by default for all table buckets. You can edit or disable this option by specifying a maintenance configuration file for your table bucket.

Editing this configuration requires the `s3tables:PutTableBucketMaintenanceConfiguration` permission.

Topics

- Unreferenced file removal
- Consideration and limitations

Unreferenced file removal

Unreferenced file removal identifies and deletes all objects that are not referenced by any table snapshots. As part of your unreferenced file removal policy, you can configure two properties: `unreferencedDays` (3 days by default) and `nonCurrentDays` (10 days by default).

For any object not referenced by your table and older than the `unreferencedDays` property, S3 marks the object as noncurrent. S3 deletes noncurrent objects after the number of days specified by the `nonCurrentDays` property.

> **Note**
>
> Deletes of noncurrent objects are permanent with no way to recover these objects.

To view or recover objects that have been marked as noncurrent, you must contact AWS Support. For information about contacting AWS Support, see Contact AWS or the AWS Support Documentation.

Unreferenced file removal determines the objects to delete from your table with reference only to that table. Any reference made to these objects outside of the table will not prevent unreferenced file removal from deleting an object.

If you disable unreferenced file removal, any in-progress jobs will not be affected. The new configuration will take effect for the next job after the configuration change. For more information, see the pricing information in the Amazon S3 pricing.

You can only configure unreferenced file removal at the table bucket level. This configuration will apply to every table in your bucket.

To configure unreferenced file removal by using the AWS CLI

The following example will set the `unreferencedDays` to 4 days and the `nonCurrentDays` to 10 days using the `PutTableBucketMaintenanceConfiguration` API.

```
aws s3tables put-table-bucket-maintenance-configuration \
   --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
   --type icebergUnreferencedFileRemoval \
   --value '{"status":"enabled","settings":{"icebergUnreferencedFileRemoval":
{"unreferencedDays":4,"nonCurrentDays":10}}}'
```

For more information, see put-table-bucket-maintenance-configuration in the *AWS CLI Command Reference*.

Consideration and limitations

To learn more about additional consideration and limits for unreferenced file removal, see the section called "Considerations and limitations".

Maintenance for tables

S3 Tables offers maintenance operations to enhance the management and performance of your individual tables. The following options are enabled by default for all tables in table buckets. You can edit or disable these by specifying maintenance configuration files for your S3 table.

Editing this configuration requires the `s3tables:GetTableMaintenanceConfiguration` and `s3tables:PutTableMaintenanceConfiguration` permissions.

> **Note**
>
> You can track S3 Tables automated maintenance operations on your tables through CloudTrail logs, for more information, see CloudTrail management events for S3 Tables maintenance.

Topics

- Compaction
- Snapshot management

- Consideration and limitations

Compaction

Compaction is configured at the table level and combines multiple smaller objects into fewer, larger objects to improve Apache Iceberg query performance. When combining objects, compaction also applies the effects of row-level deletes in your table.

Compaction is enabled by default for all tables, with a default target file size of 512MB, or a custom value you specify between 64MB to 512MB. The compacted files are written as the most recent snapshot of your table.

> **Note**
>
> Compaction is supported on Apache Parquet, Avro, and ORC file types.

Compaction Strategies

You can choose from multiple compaction strategies which can further increase query performance depending on your query patterns and table sort order.

S3 Tables supports these compaction strategies for tables:

- **Auto (Default)**
  - Amazon S3 selects the best compaction strategy based on your table sort order. This is the default compaction strategy for all tables.
  - For tables with a defined sort order in their metadata, `auto` will automatically apply `sort` compaction.
  - For tables without a sort order, `auto` will default to using `binpack` compaction.
- **Binpack**
  - Combines small files into larger files, typically targeting sizes over 100MB, while applying any pending deletes. This is the default compaction strategy for unsorted tables.
- **Sort**
  - Organizes data based on specified columns which are sorted automatically by hierarchy during compaction, improving query performance for filtered operations. This strategy is recommended when your queries frequently filter on specific columns. When you use this

strategy, S3 Tables automatically applies hierarchical sorting on columns when a `sort_order` is defined in table properties.

- **Z-order**
  - Optimizes data organization by blending multiple attributes into a single scalar value that can be used for sorting, allowing efficient querying across multiple dimensions. This strategy is recommended when you need to query data across multiple dimensions simultaneously. This strategy requires you to define a sort order in your Iceberg table properties using the `sort_order` table property.

Compaction will incur additional costs. The `z-order` and `sort` compaction strategies may incur a higher cost than `binpack`. For more information, see the pricing information in the Amazon S3 pricing.

Compaction Examples

The following examples show configurations for table compaction.

To configure the compaction target file size by using the AWS CLI

The minimum target compaction file size is 64MB; the maximum is 512MB.

The following example will change the target file size to 256MB using the `PutTableMaintenanceConfiguration` API.

```
aws s3tables put-table-maintenance-configuration \
   --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-bucket1 \
   --type icebergCompaction \
   --namespace mynamespace \
   --name testtable \
   --value='{"status":"enabled","settings":{"icebergCompaction":
{"targetFileSizeMB":256}}}'
```

For more information, see put-table-maintenance-configuration in the *AWS CLI Command Reference*.

To configure the compaction strategy by using the AWS CLI

The following example will change the compaction strategy to `sort` using the `PutTableMaintenanceConfiguration` API. When setting compaction you can choose from the following compaction strategies: `auto`,`binpack`,`sort`, or `z-order`

> **Note**
>
> To set the compaction strategy to `sort` or `z-order` you need the following prerequisites:
>
> - A sort order defined in your Iceberg table properties.
> - The `s3tables:GetTableData` permission.

```
aws s3tables put-table-maintenance-configuration \
   --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
   --type icebergCompaction \
   --namespace mynamespace \
   --name testtable \
   --value='{"status":"enabled","settings":{"icebergCompaction":
{"strategy":"sort"}}}'
```

For more information, see put-table-maintenance-configuration in the *AWS CLI Command Reference*.

To disable compaction by using the AWS CLI

The following example will disable compaction using the `PutTableMaintenanceConfiguration` API.

```
aws s3tables put-table-maintenance-configuration \
   --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
   --type icebergCompaction \
   --namespace mynamespace \
   --name testtable \
   --value='{"status":"disabled","settings":{"icebergCompaction":
{"targetFileSizeMB":256}}}'
```

For more information, see put-table-maintenance-configuration in the *AWS CLI Command Reference*.

Snapshot management

Snapshot management determines the number of active snapshots for your table. This is based on the `MinimumSnapshots` (1 by default) and `MaximumSnapshotAge` (120 hours by default). Snapshot management expires and removes table snapshots based on these configurations.

When a snapshot expires, Amazon S3 marks any objects referenced only by that snapshot as noncurrent. These noncurrent objects are deleted after the number of days specified by the `NoncurrentDays` property in your unreferenced file removal policy.

> **Note**
>
> Deletes of noncurrent objects are permanent with no way to recover these objects.

To view or recover objects that has been marked as noncurrent you must contact AWS Support. For information about contacting AWS Support, see Contact AWS or the AWS Support Documentation.

Snapshot management determines the objects to delete from your table with references only to that table. References to these objects from outside the table will not prevent snapshot management from deleting them.

> **Note**
>
> Snapshot management does not support retention values that you configure as Apache Iceberg table properties in the `metadata.json` file or through an `ALTER TABLE SET TBLPROPERTIES` SQL command. If any of the following conditions exist, snapshot management will fail for the entire table and Amazon S3 will not expire or remove any snapshots:
>
> - **User-defined tags or branches** – If any user-defined tag or branch exists on the table, snapshot management will fail for the entire table. This applies even if the tag or branch has a short retention period. To restore automated snapshot expiration, remove all user-defined tags and branches from the table.
> - **Iceberg snapshot retention table properties** – If the `history.expire.max-snapshot-age-ms` or `history.expire.min-snapshots-to-keep` property is set as an Apache Iceberg table property, snapshot management will fail for the entire table regardless of the configured value. To restore automated snapshot expiration, remove these properties:

```
ALTER TABLE mydb.mytable UNSET TBLPROPERTIES ('history.expire.max-snapshot-
age-ms');
ALTER TABLE mydb.mytable UNSET TBLPROPERTIES ('history.expire.min-snapshots-
to-keep');
```

To diagnose snapshot management failures, use the `GetTableMaintenanceJobStatus` API or run the following AWS CLI command. If snapshot management has failed, the response includes a `FAILED` status with a message that describes the cause of the failure.

```
aws s3tables get-table-maintenance-job-status \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-
demo-table-bucket \
    --namespace my_namespace \
    --name my_table
```

You can only configure snapshot management at the table level. For more information, see the pricing information in the Amazon S3 pricing.

Snapshot Management Examples

The following examples show configurations for table snapshot management.

To configure the snapshot management by using the AWS CLI

The following example will set the `MinimumSnapshots` to 10 and the `MaximumSnapshotAge` to 2500 hours using the `PutTableMaintenanceConfiguration` API.

```
aws s3tables put-table-maintenance-configuration \
--table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
--namespace my_namespace \
--name my_table \
--type icebergSnapshotManagement \
--value '{"status":"enabled","settings":{"icebergSnapshotManagement":
{"minSnapshotsToKeep":10,"maxSnapshotAgeHours":2500}}}'
```

To disable snapshot management by using the AWS CLI

The following example will disable snapshot management using the `PutTableMaintenanceConfiguration` API.

```
aws s3tables put-table-maintenance-configuration \
--table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
--namespace my_namespace \
--name my_table \
--type icebergSnapshotManagement \
--value '{"status":"disabled","settings":{"icebergSnapshotManagement":
{"minSnapshotsToKeep":1,"maxSnapshotAgeHours":120}}}'
```

For more information, see put-table-maintenance-configuration in the *AWS CLI Command Reference*.

Consideration and limitations

To learn more about additional consideration and limits for compaction and snapshot management, see the section called “Considerations and limitations”.

> **Note**
>
> S3 Tables applies the parquet's row-group-default size of 128 MB.

Record expiration for tables

By default, records in your S3 tables don't expire. To help minimize storage costs for your tables, you can enable and configure record expiration for the tables. With this option, Amazon S3 automatically removes records from a table when the records expire.

If you enable record expiration for a table, you specify the number of days to retain records in the table before the records expire. This can be any number of days ranging from 1 day through 2,147,483,647 days. For example, to retain table records for one year, specify 365 days. The records then persist for 365 days. After 365 days, the records expire and Amazon S3 automatically removes them.

You can enable and configure record expiration for AWS managed tables that store specific data sets from certain AWS services, currently Amazon S3 Storage Lens and Amazon SageMaker Catalog. Record expiration options aren't currently available for other AWS managed tables. The exception is Amazon S3 metadata journal tables. Journal tables use distinct record expiration settings that you specify at the service level. For information about configuring record expiration for this type of table, see Expiring journal table records. Note that record expiration options aren't available for S3 tables that you create.

After you enable record expiration for a table, you can disable it at any time. Amazon S3 then stops expiring and removing records from the table.

Topics

- How record expiration works
- Configuring record expiration for a table
- Monitoring record expiration for a table
- Considerations

How record expiration works

Record expiration automatically removes records from an S3 table when the records are older than the number of days that you specify in the record expiration settings for the table. To determine when records expire, Amazon S3 uses specific timestamps in the records. The choice of timestamp column derives directly from the table schema for a table. You don't need to specify which timestamp column to use. The tables are managed by AWS and Amazon S3 automatically chooses the appropriate column to use when you enable record expiration for a table.

You can enable and configure record expiration settings for AWS managed tables that store specific Amazon S3 Storage Lens metrics or specific Amazon SageMaker Catalog metadata. Record expiration options are available for the following AWS managed tables for those services:

- S3 Storage Lens – `bucket_property_metrics`, `default_activity_metrics`, `default_storage_metrics`, `expanded_prefixes_activity_metrics`, and `expanded_prefixes_storage_metrics`. To determine when records in these tables expire, Amazon S3 uses the `report_time` field in the records.
- Amazon SageMaker Catalog – ASSET. To determine when records in this table expire, Amazon S3 uses the `snapshot_time` field in the records.

After you enable record expiration for a table, Amazon S3 starts running record expiration jobs that perform the following operations for the table:

1. Identify records that are older than the specified expiration setting.
2. Create a new snapshot that excludes references to the expired records.

Removal is also based on snapshot expiration and unreferenced file removal settings in the maintenance configuration settings for the table. To learn more about these settings, see Maintenance for tables.

> **Warning**
>
> Amazon S3 expires and removes records within 24 to 48 hours after the records become eligible for expiration. Table records are removed from the latest snapshot. Data and storage for the records are removed through table maintenance operations. Table records can't be recovered after they expire.

Configuring record expiration for a table

You can enable, configure, and otherwise manage the record expiration settings for an S3 table by using the Amazon S3 console, Amazon S3 REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Before you try to perform these tasks for a table, make sure that you have the following AWS Identity and Access Management (IAM) permissions:

- `s3tables:GetTableRecordExpirationConfiguration` – This action allows you to access current record expiration settings for tables.
- `s3tables:PutTableRecordExpirationConfiguration` – This action allows you to enable, configure, and disable record expiration settings for tables.
- `s3tables:GetTableRecordExpirationJobStatus` – This action allows you to monitor the status of record expiration operations (jobs) for tables and access metrics for the operations.

The following sections explain how to enable, configure, and disable record expiration settings for a table by using the Amazon S3 console and the AWS CLI. To perform these tasks with the Amazon

S3 REST API or an AWS SDK, use the PutTableRecordExpirationConfiguration operation. For more information, see Developing with Amazon S3 in the *Amazon Simple Storage Service API Reference*.

Using the S3 console

To enable and configure record expiration settings for an S3 table by using the console, follow these steps.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. On the **Table buckets** page, choose the bucket that stores the table.
4. On the **Tables** tab, choose the table.
5. On the **Maintenance** tab, in the **Record expiration** section, choose **Edit**.
6. Under **Record expiration**, choose **Enable**.
7. For **Days after which records expire**, enter the number of days to retain records in the table. This can be any whole number ranging from 1 through 2,147,483,647. For example, to retain records for one year, enter **365**.

   > **Warning**
   >
   > As you determine the appropriate retention period for records in the table, note that records can't be recovered after they expire.

8. Choose **Save changes**.

To subsequently change the retention period, repeat the preceding steps.

To subsequently disable record expiration, repeat steps 1 through 5. Then, for step 6, choose **Disable**. When you finish, choose **Save changes**.

Using the AWS CLI

To configure and manage record expiration settings for an S3 table by using the AWS CLI, run the put-table-record-expiration-configuration command.

You can start by creating a JSON file that contains the record expiration settings to apply to the table. The following example shows the contents of a JSON file that enables record expiration for

a table. It also specifies a retention period of 30 days for records in the table. In other words, it specifies that table records should expire after 30 days.

```
{
    "status": "enabled",
        "settings": {
            "days": 30
        {
}
```

To use the preceding example, replace the *user input placeholders* with your own information.

> **Warning**
>
> As you determine the appropriate retention period for records in the table, note that records can't be recovered after they expire.

To disable record expiration for a table, specify `disabled` for the `status` field and omit the `settings` object from the file. For example:

```
{
    "status": "disabled"
}
```

After you create a JSON file with the settings to apply, run the `put-table-record-expiration-configuration` command. For the `table-arn` parameter, specify the Amazon Resource Name (ARN) of the table. For the `value` parameter, specify the name of the file that stores the settings.

For example, the following command updates the record expiration settings for a table. The settings are specified in a file named *record-expiration-config.json*.

```
aws s3tables put-table-record-expiration-configuration \
    --table-arn arn:aws:s3tables:us-east-1:123456789012:bucket/amzn-s3-demo-table-bucket/table/amzn-s3-demo-table \
    --value file://./record-expiration-config.json
```

To use the preceding example, replace the *user input placeholders* with your own information.

Monitoring record expiration for a table

To monitor the status and results of record expiration operations for your S3 tables, use the GetTableRecordExpirationJobStatus operation or, if you're using the AWS CLI, run the get-table-record-expiration-job-status command. In your request, specify the Amazon Resource Name (ARN) of the table.

For example, the following AWS CLI command retrieves the status of record expiration operations for a specific table in a table bucket. To use this example, replace the *user input placeholders* with your own information.

```
aws s3tables get-table-record-expiration-job-status \
    --table-arn arn:aws:s3tables:us-east-1:123456789012:bucket/amzn-s3-demo-table-bucket/table/amzn-s3-demo-table
```

If your request succeeds, you receive a response that provides details such as when Amazon S3 most recently ran record expiration operations for the table and the status of that run. If the most recent run succeeded, the response also includes processing metrics—for example, the number of data files and records that were removed, and the total size of the data that was removed. If errors occurred during the most recent run, the response includes a failure message that describes why the run failed.

Considerations

As you configure and manage record expiration settings for your AWS managed S3 tables, keep the following in mind:

- Record expiration is available only for certain AWS managed tables created by supported AWS services, Amazon S3 Storage Lens and Amazon SageMaker Catalog. In addition, record expiration is available only for individual tables, not entire table buckets.
- To determine when records expire, Amazon S3 uses specific timestamps in the tables. These timestamps represent when the data was created, not when Amazon S3 ingested the records in a table. The timestamp column that's used depends on the service that publishes the table: for S3 Storage Lens metrics, the `report_time` field; and, for Amazon SageMaker Catalog metadata, the `snapshot_time` field. You cannot specify which field to use because the tables are managed by AWS.

- If there are delays when data is exported to a table, records might become eligible for expiration sooner than you expect. For this reason, we recommend that you account for potential ingestion delays by adding buffer to the retention period in the expiration settings for your tables.
- Records expire and are removed within 24 to 48 hours after they become eligible for expiration. Amazon S3 doesn't expire and remove records immediately after they become eligible for expiration.
- Records cannot be recovered after they expire and are removed.

Considerations and limitations for maintenance jobs

Amazon S3 offers maintenance operations to enhance the performance of your S3 tables or table buckets. These options are file compaction, snapshot management, and unreferenced file removal. The following are limitations and consideration for these management options.

Topics

- Considerations for compaction
- Considerations for snapshot management
- Considerations for unreferenced file removal
- S3 table and table buckets maintenance operations limits and related APIs

Considerations for compaction

The following considerations apply to compaction. For more information about compaction, see the section called “Table maintenance”.

- Compaction is supported on Apache Parquet, Avro, and ORC file types.
- Compaction writes new files in Apache Parquet format by default. To compact files into Avro or ORC formats instead, set the `write.format.default` table property to `avro` or `orc`.
- Compaction doesn’t support data type: `Fixed`.
- Compaction doesn’t support compression types: `brotli`, `lz4`.
- Compaction occurs on an automated schedule. If you want to prevent charges associated with compaction you can manually disable it for a table using the PutTableMaintenanceConfiguration API operation.

> **Note**
>
> Apache Iceberg uses an optimistic concurrency model along with conflict detection to arbitrate write transactions. With optimistic concurrency, user and compaction transactions can conflict causing transactions to fail. If conflicts occur, compaction jobs will retry on failure. It is recommended that your pipelines also use retry logic to overcome transactions that fail from conflicting operations.

Considerations for snapshot management

The following considerations apply to snapshot management. For more information about snapshot management, see the section called "Table maintenance".

- Snapshots will be preserved only when both criteria are satisfied: the minimum number of snapshots to keep and the specified retention period.
- Snapshot management deletes expired snapshot metadata from Apache Iceberg, preventing time travel queries for expired snapshots and optionally deleting associated data files.
- Snapshot management does not support retention values you configure as Iceberg table properties in the `metadata.json` file or through an `ALTER TABLE SET TBLPROPERTIES` SQL command, including branch or tag-based retention. Snapshot management is disabled when you configure a branch or tag-based retention policy, or configure a retention policy on the `metadata.json` file that is longer than the values configured through the `PutTableMaintenanceConfiguration` API. In these cases S3 will not expire or remove snapshots and you will need to manually delete snapshots or remove the properties from your Iceberg table to avoid storage charges.
- Snapshot expiration does not use `unreferencedDays` (a setting for unreferenced file removal) or evaluate an object's creation time. When snapshot expiration removes the last snapshot reference to an object, Amazon S3 immediately marks the object as noncurrent, regardless of its age. Amazon S3 permanently deletes the object after the configured `nonCurrentDays` period.

Considerations for unreferenced file removal

The following considerations apply to unreferenced file removal. For more information about unreferenced file removal, see the section called "Table bucket maintenance".

- Unreferenced file removal deletes data and metadata files that are no longer referenced by Iceberg metadata if their creation time is before the retention period.
- **Delayed commits** – Workloads such as high-throughput streaming pipelines can create data objects before a commit adds references to those objects in a table snapshot. Configure `unreferencedDays` to be longer than the maximum expected time between object creation and successful commit completion, including processing delays and retries.

  If an object reaches the configured `unreferencedDays` age before its commit completes, Amazon S3 marks the object as noncurrent because no snapshot references it yet. If a later commit then references that object, queries and maintenance operations that read it fail. Amazon S3 permanently deletes the noncurrent object after the configured `nonCurrentDays` period, and the object cannot be recovered.

S3 table and table buckets maintenance operations limits and related APIs

Maintenance operation	Property	Configurable at table bucket level?	Configurable at table level?	Default value	Minimum value	Related Iceberg maintenance routine	Controlling S3 Tables API
Compaction	`targetFi` `eSizeMB`	No	Yes	512MB	64MB	`rewriteDa` `taFiles`	PutTableMaintenanceConfiguration
Snapshot management	`minimumSr` `apshots`	No	Yes	1	1	`ExpireSna` `pshots` `retainLas` `t`	PutTableMaintenanceConfiguration
Snapshot management	`maximumSr` `apshotAge`	No	Yes	120 hours	1 hour	`ExpireSna` `pshots` `expireOld` `erThan`	PutTableMaintenanceConfiguration

Maintenance operation	Property	Configurable at table bucket level?	Configurable at table level?	Default value	Minimum value	Related Iceberg maintenance routine	Controlling S3 Tables API
Unreferenced file removal	`unreferencedDays`	Yes	No	3 days	1 days	`deleteOrphanFiles`	PutTableBucketMaintenanceConfiguration
Unreferenced file removal	`nonCurrentDays`	Yes	No	10 days	1 days	N/A	PutTableBucketMaintenanceConfiguration

Note

S3 Tables applies the parquets row-group-default size of 128 MB.

Cost optimization for tables with Intelligent-Tiering

You can automatically optimize storage costs for tables by using S3 Intelligent-Tiering. The S3 Tables Intelligent-Tiering storage class automatically moves data to the most cost-effective access tier when access patterns change. When you use S3 Intelligent-Tiering, data accessed less frequently is automatically moved to lower-cost tiers, and moved back to the Frequent Access tier whenever you access it again.

All data is moved between tiers without retrieval fees, performance impact, or changes to availability. Additionally, table maintenance operations such as compaction are optimized based on access patterns, processing only actively accessed data in the Frequent Access tier while reducing maintenance costs on less frequently accessed data in lower-cost tiers.

Topics

- S3 Tables Intelligent-Tiering access tiers
- Auto-tiering behavior with S3 Intelligent-Tiering
- Specifying S3 Intelligent-Tiering as your storage class
- Monitoring storage usage

S3 Tables Intelligent-Tiering access tiers

When your table is stored in the S3 Intelligent-Tiering storage class, Amazon S3 continuously monitors access patterns and automatically moves table data between access tiers.

Tiering happens at the individual file level, so a single table can have files in different tiers based on access patterns. Table data is automatically moved to one of the following access tiers based on access patterns:

- **Frequent Access**: The default tier for all files. Files in other tiers automatically move back to the Frequent Access tier when accessed.
- **Infrequent Access**: If you do not access a file for 30 consecutive days, it moves to the Infrequent Access tier.
- **Archive Instant Access**: If you do not access a file for 90 consecutive days, it moves to the Archive Instant Access tier.

All tiers provide millisecond latency, high throughput performance, and are designed for 99.9% availability and 99.999999999% durability.

Auto-tiering behavior with S3 Intelligent-Tiering

The following actions constitute access that automatically moves files from the Infrequent Access tier or the Archive Instant Access tier back to the Frequent Access tier:

- Any read or write operations on table data or metadata files using `GetObject`, `PutObject`, or `CompleteMultipartUpload` actions
- `LoadTable` or `UpdateTable` actions using Iceberg REST API Operations
- S3 Tables replication operations

Other actions don't constitute access that automatically moves files from the Infrequent Access tier or the Archive Instant Access tier back to the Frequent Access tier.

> **Note**
>
> Files smaller than 128 KB are not eligible for auto-tiering and remain in the Frequent Access tier. Compaction may combine these files into fewer, larger objects and commit them back to your table as a new snapshot. The newly compacted files become eligible for auto-tiering if the new file is 128 KB or larger.

Table maintenance behavior

Automatic table maintenance operations performed by Amazon S3, such as snapshot management, unreferenced file removal, and record expiration, continue to run on your tables regardless of tier. Compaction runs only on files in the Frequent Access tier, optimizing performance for frequently accessed data while reducing maintenance costs on data in lower-cost tiers.

Maintenance operations do not affect the access tier of files in your table. Reads performed by maintenance operations do not cause files to change tiers. However, if a maintenance operation, such as compaction or record expiration, writes a new file, that file is created in the Frequent Access tier.

> **Note**
>
> Because compaction only processes files in the Frequent Access tier, delete operations on data in lower-cost tiers create delete files that are not automatically compacted. These delete files become eligible for compaction when the associated data files are accessed and move back to the Frequent Access tier. For tables that are not frequently accessed, you can manually run compaction using Amazon EMR to compact these delete files with their associated data files. For more information, see Maintaining tables by using compaction. You can monitor file growth in your table using Amazon CloudWatch metrics to determine when manual compaction may be beneficial.

Specifying S3 Intelligent-Tiering as your storage class

By default, all tables are created in the S3 Standard storage class and cannot be moved to S3 Intelligent-Tiering. To use S3 Intelligent-Tiering, you must specify it at table creation. You can also

set S3 Intelligent-Tiering as the default storage class for your table bucket to automatically store any new tables created there in the S3 Intelligent-Tiering storage class.

Specifying S3 Intelligent-Tiering for table buckets

You can specify S3 Intelligent-Tiering as the default storage class when creating a new table bucket by using the `storage-class-configuration` header with the `CreateTableBucket` operation.

To check the default storage class on an existing table bucket, use the `GetTableBucketStorageClass` operation. To modify the default storage class of an existing table bucket, use the `PutTableBucketStorageClass` operation.

> **Note**
>
> When you modify the default storage class on a table bucket, that setting applies only to new tables created in that bucket. The storage class for pre-existing tables is not changed.

Specifying S3 Intelligent-Tiering for tables

You can specify S3 Intelligent-Tiering as the storage class when creating a new table using the `storage-class-configuration` header with the `CreateTable` operation.

If you do not specify a storage class at table creation, the table is created in the default storage class configured on the table bucket at that time. Once a table is created, you cannot modify its storage class.

To check the default storage class on an existing table bucket, use the `GetTableBucketStorageClass` operation.

Monitoring storage usage

You can view your storage usage breakdown by access tier in the AWS Cost and Usage Reports for your account. For more information, see Creating Cost and Usage Reports in the *AWS Data Exports User Guide*.

The following usage types are available in your billing reports:

Usage Type	Unit	Granularity	Description
region-Tables-TimedStorage-INT-FA-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in the S3 Intelligent-Tiering Frequent Access of S3 Intelligent-Tiering storage
region-Tables-TimedStorage-INT-IA-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in the S3 Intelligent-Tiering Infrequent Access of S3 Intelligent-Tiering storage
region-Tables-TimedStorage-INT-AIA-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in the S3 Intelligent-Tiering Archive Instant Access of S3 Intelligent-Tiering storage
region-Tables-Requests-	Count	Hourly	The number of PUT, COPY, or POST requests on S3 Tables Intelligent-Tiering objects

Usage Type	Unit	Granularity	Description
INT-Tier1			
region-Tables-Requests-INT-Tier2	Count	Hourly	The number of GET and all other non-Tier1 requests for S3 Tables Intelligent-Tiering objects

Table namespaces

When you create tables within your Amazon S3 table bucket, you organize them into logical groupings called *namespaces*. Unlike S3 tables and table buckets, namespaces aren't resources. Namespaces are constructs that help you organize and manage your tables in a scalable manner. For example, all the tables belonging to the human resources department in a company could be grouped under a common namespace value of `hr`.

To control access to specific namespaces, you can use table bucket resource policies. For more information, see the section called “Resource-based policies”.

The following rules apply to table namespaces:

- Each namespace must be unique within a table bucket.
- You can create up to 10,000 namespaces per table bucket.
- Each table name must be unique within a namespace.
- Each table can have only one level of namespaces. Namespaces can't be nested.
- Each table belongs to a single namespace.
- You can move your tables between namespaces.

Table namespaces are referred to as databases in various AWS services and query engines. The following table maps the terminology used for S3 Tables namespaces to some common engines and services.

Service or Engine	Terminology
AWS Lake Formation	Database
AWS Glue Data Catalog	Database
Athena	Database
Spark	Namespace

Topics

- Creating a namespace
- Delete a namespace

Creating a namespace

A table namespace is a logical construct that you group tables under within an Amazon S3 table bucket. Each table belongs to a single namespace. Before creating a table in a table bucket, you must create a namespace to group tables under. You can create a namespace by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), Amazon S3 REST API, AWS SDKs, or integrated query engines.

Namespace names

The following naming rules apply to namespaces:

- Names must be between 1 and 255 characters long.
- Names can consist only of lowercase letters, numbers, and underscores (_). Underscores aren't allowed at the start or end of namespace names.
- Names must begin and end with a letter or number.
- Names must not contain hyphens (-) or periods (.).
- A namespace must be unique within a table bucket.
- Namespace names must not start with the reserved prefix `aws`.

For more information about valid namespace names, see Naming rules for tables and namespaces.

Using the S3 console and Amazon Athena

The following procedure uses the **Create table with Athena** workflow to create a namespace in the Amazon S3 console. If you don't want to also use Amazon Athena to create a table in your namespace, you can cancel the workflow after creating your namespace.

To create a namespace

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. On the **Table buckets** page, choose the bucket that you want to create a namespace in.
4. On the bucket details page, choose **Create table with Athena**.
5. In the **Create table with Athena** dialog box, choose **Create a namespace**, and then choose **Create namespace**.
6. Enter a name in the **Namespace name** field. Namespace names must be 1 to 255 characters and unique within the table bucket. Valid characters are a–z, 0–9, and underscores (_). Underscores aren't allowed at the start or end of namespace names.
7. Choose **Create namespace**.
8. If you also want to create a table, choose **Create table with Athena**. For more information about creating a table with Athena, see the section called "Using the S3 console and Amazon Athena". If you don't want to create a table right now, choose **Cancel**.

Using the AWS CLI

This example shows how to create a table namespace by using the AWS CLI. To use this example, replace the *user input placeholders* with your own information.

```
aws s3tables create-namespace \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-bucket1 \
    --namespace example_namespace
```

Using a query engine

You can create a namespace in an Apache Spark session connected to your Amazon S3 table buckets.

This example shows you how to create a table by using CREATE statements in a query engine integrated with S3 Tables. To use this example, replace the *user input placeholders* with your own information.

```
spark.sql("CREATE NAMESPACE IF NOT EXISTS s3tablesbucket.my_namespace")
```

Delete a namespace

Before you delete a table namespace from an Amazon S3 table bucket, you must delete all tables within the namespace, or move them under another namespace. You can delete a namespace by using the Amazon S3 REST API, AWS SDKs, AWS Command Line Interface (AWS CLI), or integrated query engines.

For information about the permissions required to delete a namespace, see DeleteNamespace in the *Amazon Simple Storage Service API Reference*.

Using the AWS CLI

This example shows you how to delete a table namespace by using the AWS CLI. To use this example, replace the *user input placeholders* with your own information.

```
aws s3tables delete-namespace \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-bucket1 \
    --namespace example_namespace
```

Tables in S3 table buckets

An S3 table represents a structured dataset consisting of underlying table data and related metadata. This data is stored inside a table bucket as a subresource. All tables in a table bucket are stored in the Apache Iceberg table format. Amazon S3 manages maintenance of your tables through automatic file compaction and snapshot management. For more information, see Maintenance for tables.

To make tables in your account accessible by AWS analytics services, you integrate your Amazon S3 table buckets with AWS Glue Data Catalog. This integration allows AWS analytics services such as Amazon Athena and Amazon Redshift to automatically discover and access your table data.

When you create a table, Amazon S3 automatically generates a warehouse location for the table. This is a unique S3 location that stores objects associated with the table. The following example shows the format of a warehouse location:

```
s3://63a8e430-6e0b-46f5-k833abtwr6s8tmtsycedn8s4yc3xhuse1b--table-s3
```

Within your table bucket, you can organize tables into logical groupings called namespaces. For more information, see Table namespaces.

You can rename tables, but each table has its own unique Amazon Resource Name (ARN) and unique table ID. Each table also has a resource policy attached to it. You can use this policy to manage access to the table.

Table ARNs use the following format:

```
arn:aws:s3tables:region:owner-account-id:bucket/bucket-name/table/table-id
```

By default, you can create up to 10,000 tables in a table bucket. To request a quota increase for table buckets or tables, contact Support.

Amazon S3 supports the following types of tables in table buckets:

Customer tables

Customer tables are tables that you can read and write to. You can retrieve data from these tables using integrated query engines. You can insert, update, or delete data within them by using S3 API operations or integrated query engines.

AWS tables

AWS tables are read-only tables that are generated by an AWS service on your behalf. These tables are managed by Amazon S3 and can't be modified by any IAM principal outside of Amazon S3 itself. You can retrieve information from these tables, but you can't modify the data in them. AWS tables include S3 Metadata tables, which contain metadata that's captured from the objects within an S3 general purpose bucket. For more information, see Discovering your data with S3 Metadata tables.

Topics

- Creating an Amazon S3 table

- Migrating tables to S3 Tables
- Deleting an Amazon S3 table
- Viewing details about an Amazon S3 table
- Managing table policies
- Using tags with S3 tables

Creating an Amazon S3 table

An Amazon S3 table is a subresource of a table bucket. Tables are stored in the Apache Iceberg format so that you can work with them by using query engines and other applications that support Apache Iceberg. Amazon S3 continuously optimizes your tables to help reduce storage costs and improve analytics query performance.

When you create a table, Amazon S3 automatically generates a *warehouse location* for the table. A warehouse location is a unique S3 location where you can read and write objects associated with the table. The following example shows the format of a warehouse location:

```
s3://63a8e430-6e0b-46f5-k833abtwr6s8tmtsycedn8s4yc3xhuse1b--table-s3
```

Tables have the following Amazon Resource Name (ARN) format:

```
arn:aws:s3tables:region:owner-account-id:bucket/bucket-name/table/table-id
```

By default, you can create up to 10,000 tables in a table bucket. To request a quota increase for table buckets or tables, contact Support.

You can create a table by using the Amazon S3 console, Amazon S3 REST API, AWS SDKs, AWS Command Line Interface (AWS CLI), or query engines connected to your table buckets.

When you create a table you can specify the encryption settings for that table, unless you are creating the table with Athena. If you don't specify encryption settings the table is encrypted with the default settings for the table bucket. For more information, see Specifying encryption for tables.

Prerequisites for creating tables

To create a table, you must first do the following:

- Create a table bucket.
- Create a namespace in your table bucket.
- Make sure that you have AWS Identity and Access Management (IAM) permissions for `s3tables:CreateTable` and `s3tables:PutTableData`.
- **Note**

  If you are using SSE-KMS encryption for your table, you need permissions for `s3tables:PutTableEncryption`, and `DescribeKey` permission on the chosen AWS KMS key. Additionally the AWS KMS key you use needs to grant S3 Tables permission to perform automatic table maintenance. For more information, see Permission requirements for S3 Tables SSE-KMS encryption

For information about valid table names, see Naming rules for tables and namespaces.

Important

When creating tables, make sure that you use all lowercase letters in your table names and table definitions. For example, make sure that your column names are all lowercase. If your table name or table definition contains capital letters, the table isn't supported by AWS Lake Formation or the AWS Glue Data Catalog. In this case, your table won't be visible to AWS analytics services such as Amazon Athena, even if your table buckets are integrated with AWS analytics services.

If your table definition contains capital letters, you receive the following error message when running a SELECT query in Athena: "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names."

Using the S3 console and Amazon Athena

The following procedure uses the Amazon S3 console to create a table with Amazon Athena. If you haven't already created a namespace in your table bucket, you can do so as part of this process. Before performing the following steps, make sure that you've integrated your table buckets with AWS analytics services in this Region. For more information, see the section called “Integrating S3 Tables with AWS analytics services”.

> **Note**
>
> When you create a table using Athena that table inherits the default encryption settings from the table bucket. If you want to use a different encryption type you need to create the table using another method.

To create a table

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. On the **Table buckets** page, choose the bucket that you want to create a table in.
4. On the bucket details page, choose **Create table with Athena**.
5. In the **Create table with Athena** dialog box, do one of the following:
   - Create a new namespace. Choose **Create a namespace**, and then enter a name in the **Namespace name** field. Namespace names must be 1 to 255 characters and unique within the table bucket. Valid characters are a–z, 0–9, and underscores (_). Underscores aren't allowed at the start of namespace names.
   - Choose **Create namespace**.
   - Specify an existing namespace. Choose **Specify an existing namespace within this table bucket**. Then choose either **Choose from existing namespaces** or **Enter an existing namespace name**. If you have more than 1,000 namespaces in your bucket, you must enter the namespace name if it doesn't appear in the list.
6. Choose **Create table with Athena**.
7. The Amazon Athena console opens and the Athena query editor appears. The **Catalog** field should be populated with **s3tablescatalog/** followed by the name of your table bucket, for example, **s3tablescatalog/*amzn-s3-demo-bucket***. The **Database** field should be populated with the namespace that you created or selected earlier.

   > **Note**
   >
   > If you don't see these values in the **Catalog** and **Database** fields, make sure that you've integrated your table buckets with AWS analytics services in this Region. For more information, see the section called “Integrating S3 Tables with AWS analytics services”.

8. The query editor is populated with a sample query that you can use to create a table. Modify the query to specify the table name and columns that you want your table to have.
9. When you're finished modifying the query, choose **Run** to create your table.

> **Note**
>
> - If you receive the error "Insufficient permissions to execute the query. Principal does not have any privilege on specified resource" when you try to run a query in Athena, you must be granted the necessary Lake Formation permissions on the table. For more information, see the section called “Granting Lake Formation permission on a table or database”.
> - If you receive the error "Iceberg cannot access the requested resource" when you try to run a query in Athena, go to the AWS Lake Formation console and make sure that you've granted yourself permissions on the table bucket catalog and database (namespace) that you created. Don't specify a table when granting these permissions. For more information, see the section called “Granting Lake Formation permission on a table or database”.
> - If you receive the following error message when running a SELECT query in Athena, this message is caused by having capital letters in your table name or your column names in your table definition: "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names." Make sure that your table and column names are all lowercase.

If your table creation was successful, the name of your new table appears in the list of tables in Athena. When you navigate back to the Amazon S3 console, your new table appears in the **Tables** list on the bucket details page for your table bucket after you refresh the list.

Using the AWS CLI

This example shows how to create a table with a schema by using the AWS CLI and specifying table metadata with JSON. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3tables create-table --cli-input-json file://mytabledefinition.json
```

For the `mytabledefinition.json` file, use the following example table definition. To create a partitioned table, include a `partitionSpec` in the table metadata. Each partition field references a column in your schema by its `sourceId` and applies a transform (such as `identity`, `bucket`, `truncate`, `year`, `month`, `day`, or `hour`) to produce the partition value. Because partition fields reference schema columns by ID, you must assign an explicit `id` to each field in your schema. The following example assigns IDs to the schema fields and partitions the table by the `registration_date` column using the `day` transform. To use this example, replace the *`user input placeholders`* with your own information.

```
{
    "tableBucketARN": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket",
    "namespace": "your_namespace",
    "name": "example_table",
    "format": "ICEBERG",
    "metadata": {
        "iceberg": {
            "schema": {
                "fields": [
                     {"id": 1, "name": "id", "type": "int", "required": true},
                     {"id": 2, "name": "name", "type": "string"},
                     {"id": 3, "name": "registration_date", "type": "timestamp"}
                ]
            },
            "partitionSpec": {
                "specId": 0,
                "fields": [
                    {"sourceId": 3, "transform": "day", "name": "registration_day"}
                ]
            }
        }
    }
}
```

Using a query engine

You can create a table in a supported query engine that's connected to your table buckets, such as in an Apache Spark session on Amazon EMR.

The following example shows how to create a table with Spark by using `CREATE` statements, and add table data by using `INSERT` statements or by reading data from an existing file. To create a

partitioned table, add a `PARTITIONED BY` clause to the `CREATE` statement. You can partition by one or more columns directly (identity partitioning) or by applying a partition transform such as `bucket`, `truncate`, `years`, `months`, `days`, or `hours`. The following example partitions the table by the `registration_date` column using the `days` transform. To use this example, replace the *`user input placeholders`* with your own information.

```
spark.sql(
" CREATE TABLE IF NOT EXISTS s3tablesbucket.example_namespace.`example_table` (
    id INT,
    name STRING,
    registration_date TIMESTAMP
)
USING iceberg
PARTITIONED BY (days(registration_date)) "
)
```

After you create the table, you can load data into the table. Choose from the following methods:

- Add data into the table by using the `INSERT` statement.

  ```
  spark.sql(
  """
      INSERT INTO s3tablesbucket.my_namespace.my_table
      VALUES
          (1, 'ABC', 100),
          (2, 'XYZ', 200)
  """)
  ```

- Load an existing data file.

  1. Read the data into Spark:

     ```
     val data_file_location = "Path such as S3 URI to data file"
     val data_file = spark.read.parquet(data_file_location)
     ```

  2. Write the data into an Iceberg table:

     ```
     data_file.writeTo("s3tablesbucket.my_namespace.my_table").using("Iceberg").tableProperty
      ("format-version", "2").createOrReplace()
     ```

Migrating tables to S3 Tables

S3 Tables can migrate AWS Glue-registered Apache Hive tables stored as Apache Parquet in Amazon S3 general purpose buckets to managed Apache Iceberg tables in S3 Tables. S3 Tables automatically copy your data to a new table and convert Hive data types to Iceberg types. Small files are combined into larger ones and data is compressed, so your tables are optimized for storage cost and query performance. You don't need to provision or manage any compute, and your source table remains unchanged and fully available for both reads and writes during a migration job. Migration doesn't move your source objects stored in S3 Intelligent-Tiering to the Frequent Access tier or reset their days since last access. You can migrate tables by using the Amazon S3 console, the AWS CLI, the S3 Tables API, or the S3 Tables migration skill.

For more information, see How to migrate a table to S3 Tables.

Features

Optimized destination table

Migration compresses or recompresses your data and compacts small files, so your destination table is optimized for both storage cost and query performance. You can configure the compression codec, target file size, and snapshot management settings, or use the defaults. For more information, see Setting up table migration.

Incremental sync before cutover

You don't need to stop writes to your source table. After the initial migration is complete, you can sync the table to migrate changes from your source table to keep the destination table current, and minimize cutover downtime.

Automatic Hive-to-Iceberg conversion

Migration maps your Hive schema and partitioning to an Iceberg schema and partition spec. You can either specify a destination schema to choose the columns and data types you want or let migration generate a compatible Iceberg schema based on the AWS Glue-registered schema. The migration policy controls how migration handles any data type or schema mismatches, while also automatically performing predefined data type conversions between compatible Hive and Iceberg types. For more information, see Migration policy.

Migration review, reports, and statistics

Migration produces reports and statistics that you can use to review your source table and track progress. Before copying data, the migration job writes compatibility reports that list issues to resolve. By default, the migration job continues automatically after planning. You can choose to pause and review the reports before any data is copied. After the copy, the object migration report shows the result for each source object. You can also check migration statistics at any time to see how a start or sync is progressing. For more information, see Migration reports and statistics.

For more information about table migration, see the following topics.

- How to migrate a table to S3 Tables
  - Data type mappings for table migration
  - Migration reports and statistics
- Setting up table migration
- Running a table migration

How to migrate a table to S3 Tables

S3 Tables migration creates an Apache Iceberg table from an existing AWS Glue Data Catalog Apache Hive table that's stored as Apache Parquet. The migration job validates, transforms, and optimizes your data. Your source data remains available throughout the migration.

When you start a migration, the migration job automatically creates a read-only destination table in S3 Tables, and manages it for the duration of the migration. The table remains read-only until you complete the migration so that a later sync doesn't overwrite your changes. You can delete the destination table at any time.

For more information, see the following topics.

Topics

- Start, Sync, and Complete
- What you can do during a migration
- How data is migrated
- Migration policy
- Partitioning
- Data type mappings for table migration

- Migration reports and statistics

Start, Sync, and Complete



You migrate a table in three steps:

1. **Start** – `StartTableMigration` creates a new destination table with the specified table name from the defined source table and copies the source data. You can optionally set additional configurations such as schema, compression codec, and migration policy.
2. **Sync** – `SyncTableMigration` copies only the objects and partitions that changed since the previous start or sync, even if the change is only within part of a partition. Sync is an optional step, and you can sync as many times as you need. You also use sync to resume a failed or cancelled migration.
3. **Complete** – `CompleteTableMigration` enables writes on the destination table in addition to reads, and turns on table maintenance. After you complete a migration, no further migration operations are permitted.

Only one start or sync operation can run on a destination table at a time. If a start or sync fails or is cancelled, see Data from failed or cancelled migrations.

What you can do during a migration

You do not have to alter the workloads running on your source table before or during the migration. You can read, write, and delete objects in your source table. In the destination table, you can query the data and delete the table at any time during the migration. Queries return the data that the migration job has committed so far. Writes, file compaction, and unreferenced file removal are disabled until you complete the migration. Snapshot management remains enabled and uses the snapshot management settings that you specify when you start the migration. Throughout the process, you retain control of the source and destination tables, including the ability to delete them or revoke permissions, though doing so can cause the migration job to fail.

How data is migrated

Each start and sync moves through two states: planning and migrating. `GetTableMigration` reports the current state, such as `planning`, `migrating`, or `succeeded`. To check the current state, call `GetTableMigration`. For a description of every state, see Migration states.

- **Planning state**: The migration job validates your request and identifies every source object in the source table. The outputs of the planning state are a table compatibility report and an object compatibility report, which list any issues that were detected. For more information, see Migration reports. You have the option to review the reports before any data is migrated. To review the reports before any data is copied, set `reviewMigrationPlan` to `true` when you start or sync. The migration then waits in the `pendingReview` state until you approve or cancel it.
- **Migrating state**: If `reviewMigrationPlan` is `false` or once you've approved the migration plan, the migration moves to the migrating state. In this state, the migration job reads each planned object, applies your migration policy, converts data types, recompresses and compacts the data, and commits it to the destination table. Each commit adds an Iceberg snapshot. At the end of the migrating state, the migration job writes an object migration report.

Migration policy

The migration policy determines how migration handles differences between your source data and the destination and governs data conversion rules. You set this in `dataMigration.migrationPolicies` when you call `StartTableMigration`, and subsequent sync operations use the same policy. The migration policy has two settings that control how migration handles mismatches, and cannot be changed once set. Both default to `fail`.

- `unmatchedSourceFieldsPolicy` controls source columns that aren't in the destination schema. Set it to `fail` (default) to fail the source object (the data file holding the unmatched column), which fails the migration, or `drop` to leave out the column.
- `incompatibleValuesPolicy` controls optional (or nullable) columns that can't be converted to their destination data type. Set it to `fail` (default) to fail the migration job, or `setNull` to write the destination column as null instead.

unmatchedSourceFieldsPolicy	incompatibleValuesPolicy	Source column that isn't in the destination	Incompatible column or value
fail (default)	fail (default)	Fails	Fails
fail	setNull	Fails	Written as null
drop	fail	Dropped (not migrated)	Fails
drop	setNull	Dropped (not migrated)	Written as null

A source column is required only if the destination schema marks the matching destination column as required. To specify a schema, see Setting up table migration.

The object migration report lists the objects that failed and the columns that were dropped or written as null. If any object fails, the migration job ends in the `failed` state. Failed jobs will not migrate all the data in the source table. For more information, see Troubleshooting.

Migration policy example

Your AWS Glue table `orders` has the columns `order_id` (`bigint`) and `quantity` (`int`), and you let migration infer the schema.

Orders Tables

Column	AWS Glue data type
order_id	bigint
quantity	int

`file-a.parquet` has an extra column, `audit`. `file-b.parquet` stores `quantity` as `bigint`, which is incompatible with `int`. The mismatch is handled as follows based on your migration policy.

unmatchedSourceFieldsPolicy	incompatibleValuesPolicy	file-a.parquet (extra column)	file-b.parquet (bigint in an int column)
fail	fail	Fails	Fails
fail	setNull	Fails	Migrates, with quantity written as null for this object
drop	fail	Migrates without the audit column	Fails
drop	setNull	Migrates without the audit column	Migrates, with quantity written as null for this object

Partitioning

Migration keeps your source table's partitioning. The migration job copies the objects under each partition location that's registered in AWS Glue, including objects in subdirectories. For an unpartitioned table, migration copies the objects directly under the table location. For partition requirements and limits, see Considerations and limitations.

Data type mappings for table migration

What migration transforms

A migration job applies the following transformations as it copies your data. It doesn't change your source table or its objects.

Transformation	What the migration job does
Data types	Converts each source column data type to an Iceberg-compatible data type.

Transformation	What the migration job does
Column matching	Matches source columns to destination columns by name, ignoring case, at every nesting level. If two source columns match the same destination column, the migration fails during planning.
Partitioning	Represents your Hive partitions as an Iceberg partition spec that uses identity transforms. The migration job writes the partition values into each data file.
File format and compression	Writes Apache Parquet files and recompresses your data.
File size and layout	Combines small files and splits large ones toward a target file size.

Iceberg version support

Migration creates Iceberg format version 2 tables by default. Of the data types that Iceberg version 3 adds, migration supports nanosecond timestamps (`timestamp_ns` and `timestamptz_ns`). Choose version 3 if your source has nanosecond timestamps and every engine that reads the table supports version 3.

How column data types change

Migration identifies the level of fidelity with which it can convert data from the source schema and destination schema. Migration classifies the transformations of columns and values as follows.

- **Lossless** – The data type stays the same and every value is represented exactly.
- **Widened** – The column moves to a larger data type of the same kind, and every value is represented exactly. For example, `int` becomes `long`.
- **Converted** – The column moves to a different data type that keeps every value, but some information about the data type is lost. For example, a `tinyint` column becomes `decimal(5,2)`, which keeps every value but loses the information that the column only ever held whole numbers.
- **Incompatible** – The destination data type can't hold every value of the source type. For example, `bigint` becomes `int`. Your `incompatibleValuesPolicy` determines whether the migration job fails or the column is written as null.

- **Unsupported** – The source data type has no equivalent Iceberg data type, or migration doesn't support the requested data type conversion. The migration fails under every migration policy. Migration checks data types during planning, using the AWS Glue table and a sample data file. To migrate the table, leave the column out of your destination schema and set `unmatchedSourceFieldsPolicy` to `drop`. You can't leave out a partition column.

The table compatibility report lists each column's data type change. For more information on the table compatibility report, see Migration reports and statistics.

Data type mapping for an inferred schema

If you don't specify a destination schema, migration infers one from the AWS Glue table definition, and every column is optional. The following table shows the Iceberg data type for each AWS Glue data type.

AWS Glue data type	Iceberg data type	Data type change
`boolean`	`boolean`	Lossless
`tinyint`, `smallint`	`int`	Widened
`int`	`int`	Lossless
`bigint`	`long`	Lossless
`float`	`float`	Lossless
`double`	`double`	Lossless
`decimal(p,s)`	`decimal(p,s)`	Lossless
`string`	`string`	Lossless
`varchar(n)`	`string`	Widened (the length limit isn't preserved)
`char(n)`	`string`	Converted (the length limit and trailing-space padding aren't kept)

AWS Glue data type	Iceberg data type	Data type change
`binary`	`binary`	Lossless
`date`	`date`	Lossless
`timestamp`	`timestamp` or `timestamp tz` , based on the sample object	Lossless for microseconds. Widened for milliseconds. Converted for INT96. See Timestamps.
`timestamp with local time zone`	`timestamptz`	Lossless
`array<T>`, `map<K,V>`, `struct`	`list<T>`, `map<K,V>`, `struct`	Each nested field is classified on its own.
`uniontype` , `void`, `interval_year_month` , `interval_day_time` , `decimal` with precision above 38	None	Unsupported

Data type changes in a destination schema

If you specify a destination schema, you can request the data type changes in the following table. Changes that narrow a data type (for example, `bigint` to `int`), parse text (for example, `string` to a number), or drop part of a date or timestamp are incompatible and follow your `incompatibleValuesPolicy`. Other changes aren't supported. The same rules apply when a source object stores a different data type than its AWS Glue column.

Source data type	Destination data type	Result
`int`	`long`	Widened
`float`	`double`	Widened
`decimal`	A wider `decimal`	Widened

Source data type	Destination data type	Result
`int`	`double`	Converted
`tinyint`, `smallint`	`float`, `double`	Converted
An integer type	A `decimal` that holds every digit	Converted
`decimal(p,0)`	`int`, `long`, `float`, `double`	Converted, if the digits fit (9 for `int`, 18 for `long`, 7 for `float`, 15 for `double`)
`string`	`binary`	Converted
A number, `boolean`, or `date`	`string`	Converted

Timestamps

An AWS Glue `timestamp` column can store a local date and time (Iceberg `timestamp`) or an instant in UTC (Iceberg `timestamptz`). The migration job reads a sample object to choose the data type, as the following table shows.

How the sample object stores the column	Iceberg data type	Example writers
Parquet `TIMESTAMP` adjusted to UTC (`isAdjustedToUTC` is `true`), or the older `TIMESTAMP_MILLIS` or `TIMESTAMP_MICROS` type	`timestamptz`	Apache Spark with `spark.sql.parquet.outputTimestampType` set to `TIMESTAMP_MICROS` or `TIMESTAMP_MILLIS`
Parquet `TIMESTAMP` not adjusted to UTC (`isAdjustedToUTC` is `false`)	`timestamp`	Apache Spark `TIMESTAMP_NTZ` columns
INT96	`timestamptz`	Apache Hive, and Apache Spark by default

How the sample object stores the column	Iceberg data type	Example writers
The sample object doesn't contain the column (for example, the column was added after the migration job sampled it)	`timestamp`	None

You can't change a column data type between `timestamp` and `timestamptz`, except for a column stored as INT96. If a later source object stores the column differently, that object fails. Migration widens millisecond timestamps to microseconds and drops the digits below a microsecond in INT96 timestamps. Nanosecond timestamps require Iceberg format version 3.

Nested data types

Migration maps `struct`, `array`, and `map` types to Iceberg `struct`, `list`, and `map` types, and classifies each nested field on its own.

Migration reports and statistics

Migration statistics

When you call `GetTableMigration`, the response includes two sets of statistics. Planning statistics describe the source objects that planning scanned and included. Both planning and migration statistics describe a single start or sync. A sync counts the objects that it adds or re-migrates, so an object that several syncs re-migrate is counted in each of them.

Planning statistics

Statistic	Description
`objectsScanned`	The number of source objects that planning scanned under the table location.
`objectsPlanned`	The number of source objects that planning included in the migration.
`partitionCount`	The number of AWS Glue partitions that planning included in the migration.

Migration statistics

Statistic	Description
`objectsMigrated`	The number of source objects that were copied and are available for querying.
`objectsFailedMigration`	The number of source objects that failed to migrate.
`objectsPendingMigration`	The number of planned source objects that are remaining.

Migration reports

A migration job writes the following reports that you can use to review and validate a migration:

- **Table compatibility report** – Written at the end of the planning state, before any data is copied. Lists any issues with your table definition and shows how each column's data type maps to the destination table.
- **Object compatibility report** – Written at the end of the planning state, before any data is copied. Lists every source object that was scanned in planning, whether the object is included in the migration job, and any issue that prevents it from migrating, such as an archived storage class or unsupported encryption.
- **Object migration report** – Written when a start or sync operation completes. Lists every source object that the migration job attempted, whether it migrated, how its data was transformed, and why it failed. This report is not written if the operation fails in the planning state.

If a migration fails, `failureReason` in the `GetTableMigration` response tells you which report to look at. A code of `tableCompatibilityReport`, `objectCompatibilityReport`, or `objectMigrationReport` names the report to review. For every code and how to resolve it, see Troubleshooting. For each compatibility report, `migrationOutput` also returns the number of invalid findings (`invalidFindings`) by category: `access`, `metadata`, `data`, and `internal`. Within a report, the following fields are stable and can be used for automation: states, finding categories, and failure reasons. Conversely, finding subcategories, outcomes, and messages can change or be added over time, making these unsuitable for automation. Values that the migration API returns, such as states, failure reasons, and finding categories, are stable.

Report location and structure

In start and sync operations, you specify the output location of the migration reports using the `migrationOutput` parameter. Specify one general purpose bucket. You can optionally specify a prefix for each report. If a prefix is not specified for a report, all reports are written to the root of the bucket. Under each prefix, migration writes the reports for each start and sync in a folder named for its migration ID, as in the following example:

```
amzn-s3-demo-bucket/
    your-optional-prefix/
        migration-id/
            table-compatibility-report/
            object-compatibility-report/
            object-migration-report/
```

After migration writes a report, `GetTableMigration` returns the report's object key and checksum in `migrationOutput`. For the object compatibility report and the object migration report, the key points to the report's manifest. For the table compatibility report, the key points to the report file. Each start and sync is assigned a unique migration ID to prevent overwriting reports.

Compatibility reports

Table compatibility report

The table compatibility report shows whether your table can be migrated and what happens to each column. The report is a single CSV file with a header row. Each row is one finding and has the following columns:

Column	Description
`finding_type`	Whether the finding prevents the migration, such as `invalid` for an unsupported file format, or is informational, such as for a column whose data type widens.
`finding_category`	The kind of issue: `access` for permissions, `metadata` for your table definition, `data` for values such as partition values, or `internal` for an error in S3 Tables.

Column	Description
finding_subcategory	The check that found an issue, such as glueTableLocation , or a column's data type change, such as schemaTransformation.widened .
details	A JSON object. For an issue, findingDescription explains how to resolve the issue. For a column, details shows the column path, the AWS Glue data type, the data type in the sample object, the destination data type, whether the column is a partition column, the policy that applies, and what happens to the column (outcome).

For example:

finding_type	finding_category	finding_subcategory	details
invalid	metadata	glueTableParameters	{"findingDescription":"one or more unsupported Hive Parameters entries were detected (see details.unsupportedKeys); transactional (ACID) tables ... are not supported; fix the table definition before migrating","violations":["transactional=true"],"unsupportedKeys":["transactional=true"]}
informational	metadata	schemaTransformation.widened	{"findingDescription":"the field migrates from the AWS Glue type to its destination type","fName":"quantity","fId":2,"sourceType":"smallint","sampleFileType":null,"d

finding_type	finding_category	finding_subcategory	details
			estinationType":"int","partitionField":false}
informational	metadata	schemaInference.sample	{"bucket":"amzn-s3-demo-bucket","key":"orders/dt=2026-01-15/part-0.parquet","versionId":"3HL4kqtJlcpXroDTDmJ.rmSpXd3dIbrHY"}

For a column change, `outcome` in `details` shows what happens to the column. The following table shows example values.

outcome	What it means
`migrated`	The column is migrated.
`nullForFile`	The column is written as null in each source object whose values can't be converted, because `incompatibleValuesPolicy` is `setNull`.
`dropped`	The column isn't migrated, because `unmatchedSourceFieldsPolicy` is `drop`.
`fail`	The column prevents the migration.
`excluded`	The migration job leaves the column out under every migration policy, such as an index column that pandas wrote.

Object compatibility report

The object compatibility report's manifest lists the report files and their columns. The report files are CSV files without a header row, with one row for each source object that planning scanned. The number of rows equals `objectsScanned`, and the number of rows for planned objects equals `objectsPlanned`.

The report has the following columns:

Column	Description
`bucket`	The source bucket.
`key`	The object key.
`version_id`	The version ID. Empty for an unversioned object.
`status`	Whether planning included the object in the migration, for example `planned`.
`finding_type`	`invalid` if the issue prevents migration, or `informational` if it doesn't.
`finding_category`	The kind of issue, such as `metadata` for an object property. Empty if the object has no issue.
`finding_subcategory`	The check that found the issue, such as `storageClass` , or the reason that the object isn't in the migration.
`result_message`	A JSON object with the object's size, last modified date, ETag, storage class, encryption type, and AWS KMS key ARN, and a message that describes the issue and how to resolve it.

Object migration report

The object migration report's manifest lists each results file, its checksum, the report schema, and the number of objects that migrated and failed. Each results file is a CSV file without a header row, with one row for each source object that migration attempted.

The report has the following columns:

Column	Description
`bucket`	The source bucket.
`key`	The object key.
`version_id`	The version ID. Empty for an unversioned object.

Column	Description
`status`	Whether the object migrated or failed, for example `migrated`.
`http_status_code`	An HTTP status code for the object, such as `200` for a migrated object, or `500` for a failed object that can succeed if you run another sync.
`error_code`	The error code for a failed object, such as `sourceAccessDenied` or `schemaIncompatible`. Empty for a migrated object.
`result_message`	A JSON message detailing the transformations that migration applied to the object, such as values that were widened or written as null, or columns that were dropped. For a failed object, the message explains why it failed and how to fix it.

Setting up table migration

Before you migrate a table, make sure that you have the required resources and permissions. You can also choose how migration writes your destination table.

Required resources

- **Source table** – An AWS Glue Data Catalog Apache Hive external table that stores its data as Apache Parquet in an Amazon S3 general purpose bucket.
- **Destination table bucket** – The table bucket where the migration job creates the destination table. The namespace must already exist, and the destination table does not.
- **Report bucket** – A general purpose bucket where the migration job writes the compatibility reports and the object migration report.
- **Migration role** – An AWS Identity and Access Management (IAM) role that the migration job assumes to run the migration.

For account, Region, and source table requirements, see Considerations and limitations.

Required permissions

The migration role must be able to read your source and write your destination table and reports. Grant the role the following permissions:

- **Source table metadata** – `glue:GetTable`, `glue:GetPartitions`, and `glue:GetColumnStatisticsForPartition`.
- **Source data** – `s3:ListBucket`, `s3:GetObject`, and `s3:GetObjectVersion` on the source bucket.
- **Destination table** – `s3tables:CreateTable`, `s3tables:GetTable`, `s3tables:GetTableMetadataLocation`, `s3tables:UpdateTableMetadataLocation`, `s3tables:GetTableData`, `s3tables:PutTableData`, and `s3tables:PutTableMaintenanceConfiguration`. To set tags, encryption, or a storage class, also `s3tables:TagResource`, `s3tables:PutTableEncryption`, or `s3tables:PutTableStorageClass`.
- **Reports** – `s3:PutObject` on the report bucket and prefixes.
- **Encryption** – For SSE-KMS source objects, `kms:Decrypt` on the source key. For an SSE-KMS destination table, `kms:GenerateDataKey`, `kms:Decrypt`, and `kms:DescribeKey` on the destination key, which must also allow `maintenance.s3tables.amazonaws.com`. For an SSE-KMS report bucket, `kms:GenerateDataKey` on that key. For an example IAM policy with these AWS KMS permissions, see Permission requirements for S3 Tables SSE-KMS encryption.

The role's trust policy must allow the S3 Tables migration service principal, `migration.s3tables.amazonaws.com`, to assume the role. The following example permissions policy grants the required permissions. The policy includes AWS KMS statements for SSE-KMS source objects, an SSE-KMS destination table, and an SSE-KMS report bucket. To use these examples, remove the statements for keys that you don't use and replace the placeholders with your own information.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadSourceTableMetadata",
      "Effect": "Allow",
      "Action": ["glue:GetTable", "glue:GetPartitions",
 "glue:GetColumnStatisticsForPartition"],
      "Resource": [
        "arn:aws:glue:us-east-1:111122223333:catalog",
        "arn:aws:glue:us-east-1:111122223333:database/example_database",
        "arn:aws:glue:us-
east-1:111122223333:table/example_database/example_source_table"
      ]
```

```
    },
    {
      "Sid": "ListSourceBucket",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::amzn-s3-demo-source-bucket"
    },
    {
      "Sid": "ReadSourceObjects",
      "Effect": "Allow",
      "Action": ["s3:GetObject", "s3:GetObjectVersion"],
      "Resource": "arn:aws:s3:::amzn-s3-demo-source-bucket/example-table-location/*"
    },
    {
      "Sid": "WriteReports",
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/migration-reports/*"
    },
    {
      "Sid": "CreateDestinationTable",
      "Effect": "Allow",
      "Action": [
        "s3tables:CreateTable",
        "s3tables:TagResource",
        "s3tables:PutTableEncryption",
        "s3tables:PutTableStorageClass"
      ],
      "Resource": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-
bucket"
    },
    {
      "Sid": "WriteDestinationTable",
      "Effect": "Allow",
      "Action": [
        "s3tables:GetTable",
        "s3tables:GetTableMetadataLocation",
        "s3tables:UpdateTableMetadataLocation",
        "s3tables:GetTableData",
        "s3tables:PutTableData",
        "s3tables:PutTableMaintenanceConfiguration"
      ],
      "Resource": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-
bucket/table/*"
```

```
    },
    {
      "Sid": "DecryptSourceObjects",
      "Effect": "Allow",
      "Action": "kms:Decrypt",
      "Resource": "arn:aws:kms:us-east-1:111122223333:key/source-key-id",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-source-
bucket/example-table-location/*"
        }
      }
    },
    {
      "Sid": "EncryptDestinationTable",
      "Effect": "Allow",
      "Action": ["kms:GenerateDataKey", "kms:Decrypt", "kms:DescribeKey"],
      "Resource": "arn:aws:kms:us-east-1:111122223333:key/destination-key-id",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-table-bucket/table/*"
        }
      }
    },
    {
      "Sid": "EncryptReports",
      "Effect": "Allow",
      "Action": "kms:GenerateDataKey",
      "Resource": "arn:aws:kms:us-east-1:111122223333:key/report-bucket-key-id",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-
bucket/migration-reports/*"
        }
      }
    }
  ]
}
```

The following example trust policy allows the migration service principal to assume the role only for migrations into the specified table bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "migration.s3tables.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333",
          "aws:SourceArn": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-
demo-table-bucket"
        }
      }
    }
  ]
}
```

Permissions for the identity that runs the migration

The IAM identity that runs the migration needs the following permissions:

- `s3tables:StartTableMigration` and `s3tables:ListTableMigrations` on the table bucket.
- `s3tables:SyncTableMigration`, `s3tables:GetTableMigration`, `s3tables:UpdateTableMigration`, and `s3tables:CompleteTableMigration` on the destination table.
- `iam:PassRole` on the migration role, with the condition `iam:PassedToService` set to `migration.s3tables.amazonaws.com`.
- If the destination table uses SSE-KMS, `kms:DescribeKey` and `kms:GenerateDataKey` on the table's KMS key. The key must be an enabled customer managed key.

The migration role must be in the same AWS account as the identity that starts the migration.

Destination and migration settings

You set these settings and your migration policy when you start a migration, and every sync uses them. Settings that you don't specify use the defaults in the following table.

Setting	Request parameter	Default	Supported values
Compression codec	`destinationTable.properties.compressionCodec`	`zstd`	`zstd`, `snappy`, `gzip`, `uncompressed`
Dictionary encoding	`destinationTable.properties.dictionaryEncoding`	`true`	`true`, `false`
Target file size	`dataMigration.maintenanceSettings.icebergCompaction.targetFileSizeMB`	512	64–512 MB
Compaction strategy	`dataMigration.maintenanceSettings.icebergCompaction.strategy`	`binpack`	`binpack`. After you complete the migration, you can choose another strategy in the table's maintenance configuration.
Sort order	`destinationTable.icebergSortOrder`	None	Up to five columns. Requires a destination schema that you specify.
Minimum snapshots to keep	`dataMigration.maintenanceSettings.icebergSnapshotManagement.minSnapshotsToKeep`	3	3 or more
Maximum snapshot age	`dataMigration.maintenanceSettings.icebergSnapshotManagement.maxSnapshotAgeHours`	120 hours	1 hour or more
Iceberg version	`destinationTable.properties.icebergFormatVersion`	2	Versions 2 and 3

Considerations and limitations

- Migration supports AWS Glue Data Catalog Apache Hive external tables that store data as Apache Parquet, either uncompressed or compressed with Snappy, gzip, Zstandard, Brotli, or LZ4. Other file formats (e.g. ORC, Avro, and JSON) and compression codecs (e.g. LZO) aren't supported.
- Migration doesn't support transactional (ACID) Hive tables, tables already in an open table format, such as Apache Iceberg, tables that use a storage handler, or tables that skip header or footer lines.
- Migration doesn't support AWS Glue tables that are also registered with AWS Lake Formation.
- A sync fails an object that was overwritten since the previous start or sync, even if the new object has identical content, and even in a versioned bucket.
- The source table, the destination table bucket, and the report bucket must be in the same AWS account and AWS Region. Cross-account and federated AWS Glue catalogs aren't supported.
- Every partition must be stored under the table location registered in AWS Glue. If a partition is stored elsewhere, move its data under the table location and update its location in AWS Glue.
- Partition columns of data type `binary`, `uniontype`, or a nested data type aren't supported. Tables that use Amazon Athena partition projection aren't supported until you register their partitions in AWS Glue, for example with `MSCK REPAIR TABLE`.
- Migration doesn't support source objects encrypted with customer-provided keys (SSE-C).
- Migration doesn't support objects that require a restore before access, such as S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, and the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers. Restore or copy these objects first.
- Migration doesn't copy AWS Glue table parameters, SerDe settings, column statistics, column comments, or the table description. It also doesn't migrate directory markers, empty objects, or objects whose name or folder starts with `_` or `.`, which Hive ignores.
- Migration is available in the US East (N. Virginia), US East (Ohio), US West (Oregon), Asia Pacific (Mumbai), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), Europe (Frankfurt), Europe (Ireland), and Europe (Stockholm) AWS Regions.

Running a table migration

To migrate a table by using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Table buckets**.
3. Choose the name of the destination table bucket.
4. Choose **Create table**, and then choose **Migrate existing table**.
5. For **Source table**, choose the AWS Glue database and the Apache Hive table that you want to migrate.
6. For **Destination table**, choose the namespace and enter a name for the table.
7. For **IAM role**, choose an existing role that has the required permissions, or choose **Create new IAM role**. For the permissions, see Required permissions.
8. For **Report location**, choose the general purpose bucket for the reports. (Optional) Enter a prefix.
9. Choose **Start migration**.

To sync or complete the migration, choose the destination table, and then choose **Sync** or **Complete migration**.

To migrate a table by using the AWS CLI

1. Start the migration. Specify the migration role, the destination table, the source AWS Glue table, and where to write the reports. The migration job creates the destination table in the namespace that you specify.

```
aws s3tables start-table-migration \
    --role-arn arn:aws:iam::111122223333:role/S3TablesMigrationRole \
    --destination-table '{"tableBucketARN":"arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-table-
bucket","namespace":"example_namespace","name":"example_table"}' \
    --source-type glueTable --source '{"glueTable":
{"catalogId":"111122223333","databaseName":"example_database","tableName":"example_source_t
 \
    --migration-output '{"bucket":"arn:aws:s3:::amzn-s3-demo-
bucket","prefixes":{"tableCompatibilityReport":"migration-
```

```
reports","objectCompatibilityReport":"migration-
reports","objectMigrationReport":"migration-reports"}}'
```

The response returns a `migrationId` that you use to track the migration. You can also set the following options in the start command:

- **Review the compatibility reports before copying data** – Add `--review-migration-plan`, and then approve the migration in step 2.
- **Specify the destination schema** – Add `schema` to `--destination-table`.
- **Choose migration policy** – Add `--data-migration '{"migrationPolicies":{"unmatchedSourceFieldsPolicy":"drop","incompatibleValuesPolicy":"setNull"}}'`. For more information, see Migration policy.
- **Change compression and compaction** – Add `properties` to `--destination-table`, or `maintenanceSettings` to `--data-migration`. For more information, see Destination and migration settings.

2. (Optional) If you started the migration with `--review-migration-plan`, review the compatibility reports, and then approve the migration. To approve it, set the requested state to `queued`.

```
aws s3tables update-table-migration \
    --migration-id example-migration-id \
    --requested-state queued
```

The migration moves to `queued`, and then to `migrating` when the copy begins. To stop the migration instead, set `--requested-state` to `cancelled`. You can approve a migration only in the `pendingReview` state. For what happens when you cancel a table migration, see Data from failed or cancelled migrations.

3. Monitor the migration until it reaches the `succeeded` state.

```
aws s3tables get-table-migration \
    --migration-id example-migration-id
```

The response includes the migration's state, planning and migration statistics, whether it's a start or a sync, and the location of each report after it's written. If the migration fails, the response also includes a `failureReason`. For more information, see Troubleshooting.

4. (Optional) Sync the table to copy source data that changed since the previous start or sync. Specify the migration role, the destination table ARN, and where to write the reports.

```
aws s3tables sync-table-migration \
    --role-arn arn:aws:iam::111122223333:role/S3TablesMigrationRole \
    --table-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket/table/example-table-id \
    --migration-output '{"bucket":"arn:aws:s3:::amzn-s3-demo-bucket","prefixes":{"tableCompatibilityReport":"migration-reports","objectCompatibilityReport":"migration-reports","objectMigrationReport":"migration-reports"}}'
```

   A sync uses the source, migration policy, and settings that you specified when you started the migration. For more information, see Migration policy. Each sync returns a new `migrationId`, which you monitor the same way as a start. To review the compatibility reports before a sync copies data, add `--review-migration-plan`, and then approve the sync as described in step 2.

5. Verify the destination table.

6. Complete the migration.

```
aws s3tables complete-table-migration \
    --table-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket/table/example-table-id
```

   Completing a migration can't be undone. Completing makes the destination table read-write and turns on automatic table maintenance. After you complete the migration, you can query and write to the table with any engine that supports Apache Iceberg.

Monitoring a migration

To track a start or sync, call `GetTableMigration` and check `state` and the planning and migration statistics. For each state, see Migration states. You can also view a migration's state, statistics, and reports in the Amazon S3 console.

Migration states

Each start and sync reports its progress as a state.

State	Description
`new`	S3 Tables accepted the request and are validating the source and destination.
`planning`	The migration job is identifying and checking the source objects, and writing the compatibility reports.
`pendingReview`	Planning is done, and the migration is waiting for you to approve or cancel it. Only when you set `reviewMigrationPlan` to `true`.
`queued`	The migration is approved and waiting to start copying data.
`migrating`	The migration job is copying data into the destination table and committing it.
`succeeded`	Every planned object migrated.
`failed`	The migration stopped because of an error. For the cause, see `failureReason` .
`cancelling`	You cancelled the migration, and it's stopping.
`cancelled`	The migration stopped because you cancelled it.

You can cancel a start or sync in the `new`, `planning`, `pendingReview`, `queued`, or `migrating` state.

Troubleshooting

When a migration fails, `failureReason` in the `GetTableMigration` response includes a code that tells you where to look:

- **A report found issues (`tableCompatibilityReport`, `objectCompatibilityReport`, `objectMigrationReport`)** – Review the report named in the code, and fix the issues that it lists.
- **Permissions or resources (`rolePermissions`, `sourcePermissions`, `destinationPermissions`, `resourceNotFound`, `reportBucketNotSupported`)** – Check the migration role's trust policy and permissions, the bucket and key policies, and the names,

Regions, and requirements of the resources that you specified. The `detail` field names the resource involved.

- **The source can't be migrated (`sourceNotSupported`)** – See Considerations and limitations.
- **Internal error (`internalError`)** – Try again, and contact AWS Support if the error persists.

After you fix the cause, continue as described in Data from failed or cancelled migrations.

Data from failed or cancelled migrations

When a migration fails or is cancelled, the destination table keeps the data that migration already committed. To continue, call `ListTableMigrations` and check `hasValidPlan` for each earlier start or sync:

- If the destination table exists and `hasValidPlan` is `true` for an earlier start or sync, fix the cause of any failure, and then sync the table.
- If the destination table exists but `hasValidPlan` is `false` for every earlier start or sync, fix the cause of the failure, delete the destination table, and then start a new migration.
- If the destination table doesn't exist, start a new migration.

A sync migrates the objects that didn't migrate and any objects that you added, and removes the data of objects that you deleted. You can also delete the destination table at any time. Deleting it deletes the migrated data and ends the migration. To migrate the table again, start a new migration.

Deleting an Amazon S3 table

You can delete a table by using the Amazon S3 REST API, AWS SDKs, AWS Command Line Interface (AWS CLI), or by using integrated query engines.

> **Note**
>
> S3 Tables doesn't support the `DROP TABLE` operation with `purge=false`. Some versions of Apache Spark always set this flag to `false` even when running `DROP TABLE PURGE` commands. To delete a table, you can retry `DROP TABLE` with `purge=true`, or use the S3 Tables DeleteTable REST API operation.

Important

When you delete a table, you need to know the following:

- Deleting a table is permanent and cannot be undone. Before deleting a table, make sure that you have backed up or replicated any important data.
- All data and configurations associated with the table are permanently removed.

Using the AWS CLI

This example shows how to delete a table by using the AWS CLI. To use this command, replace the *user input placeholders* with your own information.

```
aws s3tables delete-table \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket \
    --namespace example_namespace --name example_table
```

Using a query engine

You can delete a table in an Apache Spark session connected to your Amazon S3 table buckets.

This example shows how to delete a table by using the `DROP TABLE PURGE` command. To use the command, replace the *user input placeholders* with your own information.

```
spark.sql(
" DROP TABLE [IF EXISTS] s3tablesbucket.example_namespace.example_table PURGE")
```

Viewing details about an Amazon S3 table

You can view the general details of a table in a table bucket, such creation details, format and type, in the console or programmatically. You can view table encryption settings, and maintenance settings programmatically by using the S3 Tables REST API, AWS CLI or AWS SDKs.

Viewing table details

Using the AWS CLI

This example shows how to get details about a table by using the AWS CLI. To use this example, replace the *user input placeholders* with your own information.

```
aws s3tables get-table --table-bucket-arn arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-table-bucket --namespace my-namespace --
name my-table
```

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. Select your table bucket, then select your table.
4. Select the **Properties** tab.
5. (Optional) For information about the permission policy attached to the table, select **Permissions**.

Previewing table data

Using the S3 console

You can preview the data in your table directly from the Amazon S3 console using the following procedure.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. On the **Table buckets** page, choose the bucket that contains the table that you want to query.
4. Select the table you want to preview then, choose **Preview**.

> **Note**
>
> The preview shows the first 10 rows and up to 25 columns of your table. Tables over 50mb cannot be previewed.

Encryption details

For more information about table bucket encryption, see Using server-side encryption with AWS KMS keys (SSE-KMS) in table buckets.

Using the AWS CLI

This example shows how to get details about encryption settings for a table by using the AWS CLI. To use this example, replace the *user input placeholders* with your own information.

```
aws s3tables get-table-encryption --table-bucket-arn arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-table-bucket --namespace my-namespace --
name my-table
```

Maintenance details

For information about maintenance settings, see Maintenance for table buckets

Using the AWS CLI

This example shows how to get details about maintenance configuration settings for a table by using the AWS CLI. To use this example, replace the *user input placeholders* with your own information.

```
aws s3tables get-table-maintenance-configuration --table-bucket-arn
 arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket --
namespace my-namespace --name my-table
```

Managing table policies

You can add, delete, update, and view table policies for tables by using the Amazon S3 console, Amazon S3 REST API, AWS SDK and the AWS CLI. For more information, see the following topics. For more information about supported AWS Identity and Access Management (IAM) actions and condition keys for Amazon S3 Tables, see Access management for S3 Tables. For example table policies, see Resource-based policies for S3 Tables.

Adding a table policy

To add a table policy to a table, you can use the Amazon S3 REST API, AWS SDK and the AWS CLI.

Using the AWS CLI

This example shows how to view the policy attached to a table by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3tables get-table-policy \
```

```
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket1/table/tableID \
    --namespace my-namespace \
    --name my-table
```

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Amazon S3**.
3. Choose **Table buckets** and select the table bucket name that contains your table, then select your table from that bucket.
4. Choose the **Permissions** tab.
5. Under **Table policy**, Choose **Edit**.
6. In the policy editor, enter your policy JSON.
7. (Optional) Choose **Policy examples** to see sample policies that you can adapt to your needs.
8. After entering your policy, choose **Save changes**.

Viewing a table policy

To view the bucket policy attached to a table, you can use the Amazon S3 REST API, AWS SDK and the AWS CLI.

Using the AWS CLI

This example shows how to view the policy attached to a table by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

```
aws s3tables get-table-policy \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket  \
    --namespace my-namespace \
    --name my-table
```

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Amazon S3**.

3. Choose **Table buckets** and select the table bucket name that contains your table, then select your table from that bucket.
4. Choose the **Permissions** tab.

Deleting a table policy

To delete a policy attached to a table, you can use the Amazon S3 REST API, AWS SDK and the AWS CLI.

Using the AWS CLI

This example shows how to delete a table policy by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

```
aws s3tables delete-table-policy \
    --table-ARN arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-
bucket \
    --namespace your-namespace \
    --name your-table
```

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Amazon S3**.
3. Choose **Table buckets** and select the table bucket name that contains your table, then select your table from that bucket.
4. Choose the **Permissions** tab.
5. Under **Table bucket policy**, choose **Delete**.

Using tags with S3 tables

An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 tables. You can tag S3 tables when you create them or manage tags on existing tables. For general information about tags, see Tagging for cost allocation or attribute-based access control (ABAC).

> **Note**
>
> There is no additional charge for using tags on tables beyond the standard S3 API request rates. For more information, see Amazon S3 pricing.

Common ways to use tags with tables

Use tags on your S3 tables for:

1. **Cost allocation** – Track storage costs by table tag in AWS Billing and Cost Management. For more information, see Using tags for cost allocation.
2. **Attribute-based access control (ABAC)** – Scale access permissions and grant access to S3 tables based on their tags. For more information, see Using tags for ABAC.

> **Note**
>
> You can use the same tags for both cost allocation and access control.

ABAC for S3 tables

Amazon S3 tables support attribute-based access control (ABAC) using tags. Use tag-based condition keys in your AWS organizations, AWS Identity and Access Management (IAM), and S3 table policies. ABAC in Amazon S3 supports authorization across multiple AWS accounts.

In your IAM policies, you can control access to S3 tables based on the table's tags by using the `s3tables:TableBucketTag/tag-key` condition key or the AWS global condition keys: `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys`.

aws:ResourceTag/key-name

Use this condition key to compare the tag key-value pair that you specify in the policy with the key-value pair attached to the resource. For example, you could require that access to a table is allowed only if the table has the tag key `Department` with the value `Marketing`.

This condition key applies to table actions performed using the Amazon S3 Console, the AWS Command Line Interface (CLI), S3 APIs, or the AWS SDKs,.

For an example policy, see 1.1 - table policy to restrict operations on the table using tags.

For additional example policies and more information, see Controlling access to AWS resources in the *AWS Identity and Access Management User Guide*.

> **Note**
>
> For actions performed on tables, this condition key acts on the tags applied to the table and not on the tags applied to the table bucket containing the table. Use the `s3tables:TableBucketTag/tag-key` instead if you would like your ABAC policies to act on the tags of the table bucket when performaing table actions.

aws:RequestTag/key-name

Use this condition key to compare the tag key-value pair that was passed in the request with the tag pair that you specify in the policy. For example, you could check whether the request to tag a table includes the tag key `Department` and that it has the value `Accounting`.

This condition key applies when tag keys are passed in a `TagResource` or `CreateTable` API operation request, or when tagging or creating a table with tags using the Amazon S3 Console, the AWS Command Line Interface (CLI), or the AWS SDKs.

For an example policy, see 1.2 - IAM policy to create or modify tables with specific tags.

For additional example policies and more information, see Controlling access during AWS requests in the *AWS Identity and Access Management User Guide*.

aws:TagKeys

Use this condition key to compare the tag keys in a request with the keys that you specify in the policy to define what tag keys are allowed for access. For example, to allow tagging during the `CreateTable` action, you must create a policy that allows both the `s3tables:TagResource` and `s3tables:CreateTable` actions. You can then use the `aws:TagKeys` condition key to enforce that only specific tags are used in the `CreateTable` request.

This condition key applies when tag keys are passed in a `TagResource`, `UntagResource`, or `CreateTable` API operations or when tagging, untagging, or creating a table with tags using the AWS Command Line Interface (CLI), or the AWS SDKs.

For an example policy, see 1.3 - IAM policy to control the modification of tags on existing resources maintaining tagging governance.

For additional example policies and more information, see Controlling access based on tag keys in the *AWS Identity and Access Management User Guide*.

s3tables:TableBucketTag/tag-key

Use this condition key to grant permissions to specific data in table buckets using tags. This condition key acts, on the most part, on the tags assigned to the table bucket for all S3 tables actions. Even when you create a table with tags, this condition key acts on the tags applied to the table bucket that contains that table. The exceptions are:

- When you create a table bucket with tags, this condition key acts on the tags in the request.

For an example policy, see 1.4 - Using the s3tables:TableBucketTag condition key.

Example ABAC policies for tables

See the following example ABAC policies for Amazon S3 tables.

> **Note**
>
> If you have an IAM or S3 Tables resource-based policy that restricts IAM users and IAM roles based on principal tags, you must attach the same principal tags to the IAM role that Lake Formation uses to access your Amazon S3 data (for example, LakeFormationDataAccessRole) and grant this role the necessary permissions. This is required for your tag-based access control policy to work correctly with your S3 Tables analytics integration.

1.1 - table policy to restrict operations on the table using tags

In this table policy, the specified IAM principals (users and roles) can perform the `GetTable` action only if the value of the table's `project` tag matches the value of the principal's `project` tag.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGetTable",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
```

```
      },
      "Action": "s3tables:GetTable",
      "Resource": "arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket/my_example_tab;e",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        }
      }
    }
  ]
}
```

1.2 - IAM policy to create or modify tables with specific tags

In this IAM policy, users or roles with this policy can only create S3 tables if they tag the table with the tag key `project` and tag value `Trinity` in the table creation request. They can also add or modify tags on existing S3 tables as long as the `TagResource` request includes the tag key-value pair `project:Trinity`. This policy does not grant read, write, or delete permissions on the tables or its objects.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateTableWithTags",
      "Effect": "Allow",
      "Action": [
        "s3tables:CreateTable",
        "s3tables:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/project": [
            "Trinity"
          ]
        }
      }
    }
  ]
}
```

1.3 - IAM policy to control the modification of tags on existing resources maintaining tagging governance

In this IAM policy, IAM principals (users or roles) can modify tags on a table only if the value of the table's `project` tag matches the value of the principal's `project` tag. Only the four tags `project`, `environment`, `owner`, and `cost-center` specified in the `aws:TagKeys` condition keys are permitted for these tables. This helps enforce tag governance, prevents unauthorized tag modifications, and keeps the tagging schema consistent across your tables.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceTaggingRulesOnModification",
      "Effect": "Allow",
      "Action": [
        "s3tables:TagResource",
        "s3tables:UntagResource"
      ],
      "Resource": "arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-
bucket/my_example_table",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "project",
            "environment",
            "owner",
            "cost-center"
          ]
        }
      }
    }
  ]
}
```

1.4 - Using the s3tables:TableBucketTag condition key

In this IAM policy, the condition statement allows access to the table bucket's data only if the table bucket has the tag key `Environment` and tag value `Production`. The

`s3tables:TableBucketTag/<tag-key>` differs from the `aws:ResourceTag/<tag-key>` condition key because, in addition to controlling access to table buckets depending on their tags, it allows you to control access to tables based on the tags on their parent table bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToSpecificTables",
      "Effect": "Allow",
      "Action": "*",
      "Resource": "arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket/*",
      "Condition": {
        "StringEquals": {
          "s3tables:TableBucketTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Managing tags for tables

You can add or manage tags for S3 tables using the Amazon S3 Console, the AWS Command Line Interface (CLI), the AWS SDKs, or using the S3 APIs: TagResource, UntagResource, and ListTagsForResource. For more information, see:

Topics

- Creating tables with tags
- Adding a tag to a table
- Viewing table tags
- Deleting a tag from a table

Creating tables with tags

You can tag Amazon S3 tables when you create them. There is no additional charge for using tags on tables beyond the standard S3 API request rates. For more information, see Amazon S3 pricing. For more information about tagging tables, see Using tags with S3 tables.

Permissions

To create a table with tags, you must have the following permissions:

- `s3tables:CreateTable`
- `s3tables:TagResource`

Troubleshooting errors

If you encounter an error when attempting to create a table with tags, you can do the following:

- Verify that you have the required Permissions to create the table and apply a tag to it.
- Check your IAM user policy for any attribute-based access control (ABAC) conditions. Your policy may require you to tag your tables with only specific tag keys and values. For more information about ABAC and example table ABAC policies, see ABAC for S3 tables.

Steps

You can create a table with tags applied by using the AWS Command Line Interface (AWS CLI), the Amazon S3 Tables REST API, and the AWS SDKs.

Using the REST API

For information about the Amazon S3 Tables REST API support for creating a table with tags, see the following section in the *Amazon Simple Storage Service API Reference*:

- CreateTable

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to create a table with tags by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

When you create a table you must provide configuration details. For more information, see Creating an Amazon S3 table. You must also name the table with a name that follows the table naming convention. For more information see Amazon S3 table bucket, table, and namespace naming rules.

Request:

```
aws --region us-west-2 \
s3tables create-table \
--endpoint https://ufwae60e2k.execute-api.us-west-2.amazonaws.com/personal/ \
--table-bucket-arn arn:aws:s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket
--tags '{"Department":"Engineering"}' \
--name my_table_abc \
--namespace my_namesapce_123a \
--format ICEBERG
```

Adding a tag to a table

You can add tags to Amazon S3 tables and modify these tags. For more information about tagging tables, see Using tags with S3 tables.

Permissions

To add a tag to a table, you must have the following permission:

- `s3tables:TagResource`

Troubleshooting errors

If you encounter an error when attempting to add a tag to a table, you can do the following:

- Verify that you have the required Permissions to add a tag to a table.
- If you attempted to add a tag key that starts with the AWS reserved prefix `aws:`, change the tag key and try again.
- The tag key is required. Also, make sure that the tag key and the tag value do not exceed the maximum character length and do not contain restricted characters. For more information, see Tagging for cost allocation or attribute-based access control (ABAC).

Steps

You can add tags to tables by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 Tables REST API, and AWS SDKs.

Using the S3 console

To add tags to a table using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. Choose the table bucket name.
4. Choose the table name.
5. Choose the **Properties** tab.
6. Scroll to the **Tags** section and choose **Add new Tag**.
7. This opens the **Add Tags** page. You can enter up to 50 tag key value pairs.
8. If you add a new tag with the same key name as an existing tag, the value of the new tag overrides the value of the existing tag.
9. You can also edit the values of existing tags on this page.
10. After you have added the tag(s), choose **Save changes**.

Using the REST API

For information about the Amazon S3 REST API support for adding tags to a table, see the following section in the *Amazon Simple Storage Service API Reference*:

- TagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to add tags to a table by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws --region us-west-2 \
s3tables tag-resource \
--resource-arn arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-
bucket/table/my_example_table \
```

```
--tags '{"Department":"engineering"}'
```

Viewing table tags

You can view or list tags applied to Amazon S3 tables. For more information about tags, see Using tags with S3 tables.

Permissions

To view tags applied to a table, you must have the following permission:

- `s3tables:ListTagsForResource`

Troubleshooting errors

If you encounter an error when attempting to list or view the tags of a table, you can do the following:

- Verify that you have the required Permissions to view or list the tags of the table.

Steps

You can view tags applied to tables by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To view tags applied to a table using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. Choose the Table bucket name.
4. Choose the table name within the Table bucket.
5. Choose the **Properties** tab.
6. Scroll to the **Tags** section to view all of the tags applied to the table.
7. The **Tags** section shows the **User-defined tags** by default. You can select the **AWS-generated tags** tab to view tags applied to your table by AWS services.

Using the REST API

For information about the Amazon S3 REST API support for viewing the tags applied to a table, see the following section in the *Amazon Simple Storage Service API Reference*:

- ListTagsforResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to view tags applied to a table. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws --region us-west-2 \
s3tables list-tags-for-resource \
--resource-arn arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket/table/my_example_table
```

Deleting a tag from a table

You can remove tags from Amazon S3 tables. For more information about tagging tables, see Using tags with S3 tables.

> **Note**
>
> If you delete a tag and later learn that it was being used to track costs or for access control, you can add the tag back to the table.

Permissions

To delete a tag from a table, you must have the following permission:

- `s3tables:UntagResource`

Troubleshooting errors

If you encounter an error when attempting to delete a tag from a table, you can do the following:

- Verify that you have the required Permissions to delete a tag from a table.

Steps

You can delete tags from tables by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 Tables REST API, and AWS SDKs.

Using the S3 console

To delete tags from a table using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. Choose the table bucket name.
4. Choose the table name.
5. Choose the **Properties** tab.
6. Scroll to the **Tags** section and select the checkbox next to the tag or tags that you would like to delete.
7. Choose **Delete**.
8. The **Delete user-defined tags** pop-up appears and asks you to confirm the deletion of the tag or tags you selected.
9. Choose **Delete** to confirm.

Using the REST API

For information about the Amazon S3 REST API support for deleting tags from a table, see the following section in the *Amazon Simple Storage Service API Reference*:

- UnTagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to delete tags from a table by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws --region us-west-2 \
s3tables untag-resource \
--resource-arn arn:aws::s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket/table/my_example_table \
--tag-keys '["department"]'
```

Accessing table data

There are multiple ways to access tables in Amazon S3 table buckets. You can integrate tables with AWS analytics services using AWS Glue Data Catalog, or access tables directly using the Amazon S3 Tables Iceberg REST endpoint or the Amazon S3 Tables Catalog for Apache Iceberg. The access method you use will depend on your catalog setup, governance model, and access control needs. The following is an overview of these access methods.

AWS Glue Data Catalog integration

This is the recommended access method for working with tables in S3 table buckets. This integration gives you a unified view of your data estate across multiple AWS analytics services through the AWS Glue Data Catalog. After integration, you can query tables using services such as Athena and Amazon Redshift. Access to tables is managed using IAM permissions. To access tables using this integration, the IAM identity you use needs access to your S3 Tables resources and actions, AWS Glue Data Catalog objects, and the query engine you're using. For more information, see Access management for S3 Tables.

Direct access

Use this method if you need to work with AWS Partner Network (APN) catalog implementations, custom catalog implementations, or if you only need to perform basic read/write operations on tables within a single table bucket. Access to tables is managed using IAM permissions. To access tables, the IAM identity you use needs access to your table resources and S3 Tables actions. For more information, see Access management for S3 Tables.

Accessing tables through the AWS Glue Data Catalog integration

You can integrate S3 table buckets with AWS Glue Data Catalog to access tables from AWS analytics services, such as Amazon Athena, Amazon Redshift, and Quick. The integration populates

the AWS Glue Data Catalog with your table resources, and federates access to those resources. For more information on integrating, see Integrating Amazon S3 Tables with AWS analytics services.

The following AWS analytics services can access tables through this integration:

- Amazon Athena
- Amazon Redshift
- Amazon EMR
- Quick
- Amazon Data Firehose
- AWS Glue ETL
- Querying S3 Tables with SageMaker Unified Studio

Accessing tables using the AWS Glue Iceberg REST endpoint

Once your S3 table buckets are integrated with AWS Glue Data Catalog, you can also use the AWS Glue Iceberg REST endpoint to connect to S3 tables from third-party query engines that support Iceberg. For more information, see Accessing Amazon S3 tables using the AWS Glue Iceberg REST endpoint.

We recommend using the AWS Glue Iceberg REST endpoint when you want to access tables from Spark, PyIceberg, or other Iceberg-compatible clients.

The following clients can access tables directly through the AWS Glue Iceberg REST endpoint:

- Any Iceberg client, including Spark, PyIceberg, and more.

Accessing tables directly

You can access tables directly from open source query engines through methods that bridge S3 Tables management operations to your Apache Iceberg analytics applications. There are two direct access methods: the Amazon S3 Tables Iceberg REST endpoint or the Amazon S3 Tables Catalog for Apache Iceberg. The REST endpoint is recommended.

We recommend direct access if you access tables in self-managed catalog implementations, or only need to perform basic read/write operations on tables in a single table bucket. For other access scenarios, we recommend the AWS Glue Data Catalog integration.

Direct access to tables is managed through either IAM identity-based policies or resource-based policies attached to tables and table buckets.

Accessing tables through the Amazon S3 Tables Iceberg REST endpoint

You can use the Amazon S3 Tables Iceberg REST endpoint to access your tables directly from any Iceberg REST compatible clients through HTTP endpoints, for more information, see Accessing tables using the Amazon S3 Tables Iceberg REST endpoint.

The following AWS analytics services and query engines can access tables directly using the Amazon S3 Tables Iceberg REST endpoint:

Supported query engines

- Any Iceberg client, including Spark, PyIceberg, and more.
- Amazon EMR
- AWS Glue ETL

Accessing tables directly through the Amazon S3 Tables Catalog for Apache Iceberg

You can also access tables directly from query engines like Apache Spark by using the S3 Tables client catalog, for more information, see Accessing Amazon S3 tables with the Amazon S3 Tables Catalog for Apache Iceberg. However, S3 recommends using the Amazon S3 Tables Iceberg REST endpoint for direct access because it supports more applications, without requiring language or engine-specific code.

The following query engines can access tables directly using the client catalog:

- Apache Spark

Amazon S3 Tables integration with AWS analytics services overview

To make tables in your account accessible by AWS analytics services, you integrate your Amazon S3 table buckets with AWS Glue Data Catalog. This integration allows AWS analytics services to automatically discover and access your table data. You can use this integration to work with your tables in these services:

- Amazon Athena
- Amazon Redshift
- Amazon EMR
- Quick
- Amazon Data Firehose

> **Note**
>
> This integration uses AWS Glue and AWS Lake Formation services and might incur AWS Glue request and storage costs. For more information, see AWS Glue Pricing. Additional pricing applies for running queries on your S3 tables. For more information, see pricing information for the query engine that you're using.

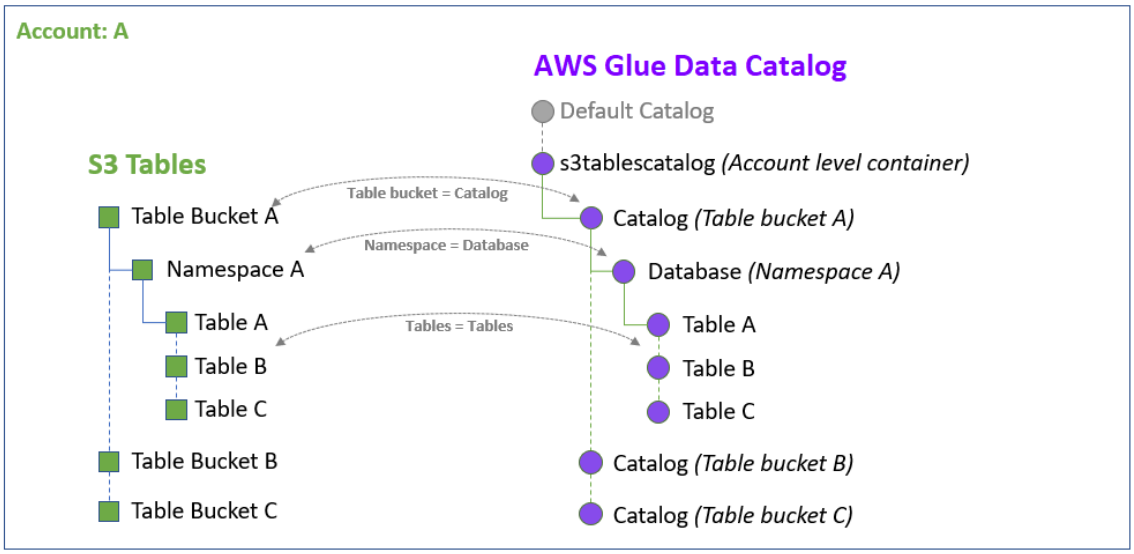
How the integration works

When you integrate S3 Tables with the AWS analytics services, Amazon S3 adds the catalog named `s3tablescatalog` to the AWS Glue Data Catalog in the current Region. Adding the `s3tablescatalog` allows all your table buckets, namespaces, and tables to be populated in the Data Catalog.

> **Note**
>
> These actions are automated through the Amazon S3 console. If you perform this integration programmatically, you must manually take these actions.

You integrate your table buckets once per AWS Region. After the integration is completed, all current and future table buckets, namespaces, and tables are added to the AWS Glue Data Catalog in that Region.

The following illustration shows how the `s3tablescatalog` catalog automatically populates table buckets, namespaces, and tables in the current Region as corresponding objects in the Data Catalog. Table buckets are populated as subcatalogs. Namespaces within a table bucket are populated as databases within their respective subcatalogs. Tables are populated as tables in their respective databases.



After integrating with Data Catalog, you can create Apache Iceberg tables in table buckets and access them through AWS analytics engines such as Amazon Athena, Amazon EMR, as well as third-party analytics engines.

How permissions work

We recommend integrating your table buckets with AWS analytics services so that you can work with your table data across services that use the AWS Glue Data Catalog as a metadata store. Once the integration is enabled, you can use AWS Identity and Access Management (IAM) permissions to grant access to S3 Tables resources and their associated Data Catalog objects.

Make sure that you follow the steps in Integrating S3 Tables with AWS analytics services so that you have the appropriate permissions to access the AWS Glue Data Catalog and your table resources, and to work with AWS analytics services.

Regions supported

S3 Tables integration with AWS analytics services uses AWS Glue Data Catalog with IAM-based access controls in the following regions. In all other regions, the integration also requires AWS Lake Formation.

- US East (N. Virginia)
- US East (Ohio)
- US West (N. California)
- US West (Oregon)

- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Taipei)
- Asia Pacific (Tokyo)
- Asia Pacific (Seoul)
- Asia Pacific (Osaka)
- Asia Pacific (Mumbai)
- Asia Pacific (Hyderabad)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Jakarta)
- Asia Pacific (Melbourne)
- Asia Pacific (Malaysia)
- Asia Pacific (New Zealand)
- Asia Pacific (Thailand)
- Canada (Central)
- Canada West (Calgary)
- Europe (Frankfurt)
- Europe (Zurich)
- Europe (Stockholm)
- Europe (Milan)
- Europe (Spain)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Israel (Tel Aviv)
- Mexico (Central)
- South America (São Paulo)

Next steps

- Integrating S3 Tables with AWS analytics services
- Create a namespace
- Create a table

Integrating Amazon S3 Tables with AWS analytics services

This topic covers the prerequisites and procedures needed to integrate your Amazon S3 table buckets with AWS analytics services. For an overview of how the integration works, see S3 Tables integration overview.

> **Note**
>
> This integration uses the AWS Glue Data Catalog and might incur AWS Glue request and storage costs. For more information, see AWS Glue Pricing.
> Additional pricing applies for running queries on S3 Tables. For more information, see pricing information for the query engine that you're using.

Prerequisites for integration

The following prerequisites are required to integrate table buckets with AWS analytics services:

- Create a table bucket.
- Add the following AWS Glue permissions to your AWS Identity and Access Management (IAM) principal:
  - `glue:CreateCatalog` which is required to create `s3tablescatalog` federated catalog in the Data Catalog
  - `glue:PassConnection` grants the calling principal the right to delegate `aws:s3tables` connection creation to Amazon S3 service.
- Update to the latest version of the AWS Command Line Interface (AWS CLI).

Important

When creating tables, make sure that you use all lowercase letters in your table names and table definitions. For example, make sure that your column names are all lowercase. If your table name or table definition contains capital letters, the table isn't supported by AWS Lake Formation or the AWS Glue Data Catalog. In this case, your table won't be visible to AWS analytics services such as Amazon Athena, even if your table buckets are integrated with AWS analytics services.
If your table definition contains capital letters, you receive the following error message when running a SELECT query in Athena: "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names."

Integrating table buckets with AWS analytics services

You can integrate table buckets with Data Catalog and AWS analytics services using IAM access controls by default, or optionally use Lake Formation access controls.

When you integrate using IAM access controls, you require IAM privileges to access Amazon S3 table buckets and tables, Data Catalog objects, and the query engine you're using. If you choose to integrate using Lake Formation, then both IAM access controls and Lake Formation grants determine the access to Data Catalog resources. Please refer to *AWS Lake Formation Developer Guide* to learn more about Lake Formation integration.

The following sections describe how you could use Amazon S3 management console or AWS CLI to configure the integration with IAM access controls.

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. Choose **Create table bucket**.

   The **Create table bucket** page opens.
4. Enter a **Table bucket name** and make sure that the **Enable integration** checkbox is selected.
5. Choose **Create table bucket**. Amazon S3 will attempt to automatically integrate your table buckets in that Region.

Using the AWS CLI

To integrate table buckets with IAM access controls using the AWS CLI

The following steps show how to use the AWS CLI to integrate table buckets. To use these steps, replace the *user input placeholders* with your own information.

1. Create a table bucket.

```
aws s3tables create-table-bucket \
--region us-east-1 \
--name amzn-s3-demo-table-bucket
```

2. Create a file called `catalog.json` that contains the following catalog:

```
{
   "Name": "s3tablescatalog",
   "CatalogInput": {
      "FederatedCatalog": {
          "Identifier": "arn:aws:s3tables:us-east-1:111122223333:bucket/*",
          "ConnectionName": "aws:s3tables"
       },
       "CreateDatabaseDefaultPermissions":[
       {
                "Principal": {
                    "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                },
                "Permissions": ["ALL"]
            }
       ],
       "CreateTableDefaultPermissions":[
       {
                "Principal": {
                    "DataLakePrincipalIdentifier": "IAM_ALLOWED_PRINCIPALS"
                },
                "Permissions": ["ALL"]
            }
       ],
       "AllowFullTableExternalDataAccess": "True"
   }
}
```

Create the `s3tablescatalog` catalog by using the following command. Creating this catalog populates the AWS Glue Data Catalog with objects corresponding to table buckets, namespaces, and tables.

```
aws glue create-catalog \
--region us-east-1 \
--cli-input-json file://catalog.json
```

3. Verify that the `s3tablescatalog` catalog was added in AWS Glue by using the following command:

```
aws glue get-catalog --catalog-id s3tablescatalog
```

Migrating to the updated integration process

The AWS analytics services integration process has been updated to use IAM permissions by default. If you've already set up the integration, you can continue to use your current integration. However, if you want to change your existing integration to use IAM permissions instead, see *AWS Lake Formation Developer Guide*. You can also redo the integration to delete your existing setup in AWS Glue Data Catalog and AWS Lake Formation and re-run the integration. This will remove all existing Lake Formation grants and associated access permissions to the `s3tablescatalog`.

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator. For more information about how to create a data lake administrator, see Create a data lake administrator in the *AWS Lake Formation Developer Guide*.
2. Delete your `s3tablescatalog` catalog by doing the following:
   - In the left navigation pane, choose **Catalogs**.
   - Select the option button next to the `s3tablescatalog` catalog in the **Catalogs** list. On the **Actions** menu, choose **Delete**.
3. Deregister the data location for the `s3tablescatalog` catalog by doing the following:
   - In the left navigation pane, go to the **Administration** section, and choose **Data lake locations**.
   - Select the option button next to the `s3tablescatalog` data lake location, for example, `s3://tables:region:account-id:bucket/*`.

- On the **Actions** menu, choose **Remove**.
- In the confirmation dialog box that appears, choose **Remove**.

4. Now that you've deleted your `s3tablescatalog` catalog and data lake location, you can follow the steps to integrate your table buckets with AWS analytics services by using the updated integration process.

> **Note**
>
> If you want to work with SSE-KMS encrypted tables in integrated AWS analytics services, the role you use needs to have permission to use your AWS KMS key for encryption operations. For more information, see Granting IAM principals permissions to work with encrypted tables in integrated AWS analytics services.

Next steps

- Create a namespace.
- Create a table.

Accessing Amazon S3 tables using the AWS Glue Iceberg REST endpoint

Once your S3 table buckets are integrated with the AWS Glue Data Catalog you can use the AWS Glue Iceberg REST endpoint to connect to your S3 tables from Apache Iceberg-compatible clients, such as PyIceberg or Spark. The AWS Glue Iceberg REST endpoint implements the Iceberg REST Catalog Open API specification which provides a standardized interface for interacting with Iceberg tables. To access S3 tables using the endpoint you need to configure permissions through a combination of IAM policies and AWS Lake Formation grants. The following sections explain how to set up access, including creating the necessary IAM role, defining the required policies, and establishing Lake Formation permissions for both database and table-level access.

For an end to end walkthrough using PyIceberg, see Access data in Amazon S3 Tables using PyIceberg through the AWS Glue Iceberg REST endpoint.

Prerequisites

- Integrate your table buckets with AWS analytics services
- Create a table namespace

- Have access to a data lake administrator account

Authentication methods

The AWS Glue Iceberg REST endpoint supports two authentication methods for reaching your Amazon S3 tables:

- **AWS Signature Version 4 (SigV4)** – The default for AWS SDKs and AWS native analytics engines. See AWS Signature Version 4 for API requests.
- **OAuth 2.0 bearer token** – For external integration partners and third-party query engines that authenticate with an OAuth access token obtained from AWS Sign-In. See Authentication in Connecting to the Data Catalog using AWS Glue Iceberg REST endpoint, and Configure OAuth access to AWS Glue Iceberg REST Catalog in the *AWS Sign-In User Guide*.

Create an IAM role for your client

To access tables through AWS Glue endpoints, you need to create an IAM role with permissions to AWS Glue and Lake Formation actions. This procedure explains how to create this role and configure its permissions.

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policies**.
3. Choose **Create a policy**, and choose **JSON** in policy editor.
4. Add the following inline policy that grants permissions to access AWS Glue and Lake Formation actions:

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "glue:GetCatalog",
                "glue:GetDatabase",
                "glue:GetDatabases",
```

```
                "glue:GetTable",
                "glue:GetTables",
                "glue:CreateTable",
                "glue:UpdateTable"
            ],
            "Resource": [
                "arn:aws:glue:us-east-1:111122223333:catalog",
                "arn:aws:glue:us-east-1:111122223333:catalog/
s3tablescatalog",
                "arn:aws:glue:us-east-1:111122223333:catalog/s3tablescatalog/
amzn-s3-demo-table-bucket",
                "arn:aws:glue:us-east-1:111122223333:table/s3tablescatalog/
amzn-s3-demo-table-bucket/<namespace>/*",
                "arn:aws:glue:us-east-1:111122223333:database/
s3tablescatalog/amzn-s3-demo-table-bucket/<namespace>"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
            "Resource": "*"
        }
    ]
}
```

5. After you create the policy, create an IAM role and choose **Custom trust policy** as the **Trusted entity type**.
6. Enter the following for the **Custom trust policy**.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/Admin_role"
            },
            "Action": "sts:AssumeRole",
```

```
            "Condition": {}
        }
    ]
}
```

Define access in Lake Formation

Lake Formation provides fine-grained access control for your data lake tables. When you integrated your S3 bucket with the AWS Glue Data Catalog, your tables were automatically registered as resources in Lake Formation. To access these tables, you must grant specific Lake Formation permissions to your IAM identity, in addition to its IAM policy permissions.

The following steps explain how to apply Lake Formation access controls to allow your Iceberg client to connect to your tables. You must sign in as a data lake administrator to apply these permissions.

Allow external engines to access table data

In Lake Formation, you must enable full table access for external engines to access data. This allows third-party applications to get temporary credentials from Lake Formation when using an IAM role that has full permissions on the requested table.

1. Open the Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator.
2. In the navigation pane under **Administration**, choose **Application integration settings**.
3. Select **Allow external engines to access data in Amazon S3 locations with full table access**. Then choose **Save**.

Grant Lake Formation permissions on your table resources

Next, grant Lake Formation permissions to the IAM role you created for your Iceberg-compatible client. These permissions will allow the role to create and manage tables in your namespace. You need to provide both database and table-level permissions. For more information, see Granting Lake Formation permission on a table or database.

Set up your environment to use the endpoint

After you have setup the IAM role with the permissions required for table access you can use it to run Iceberg clients from your local machine by configuring the AWS CLI with your role, using the following command:

```
aws sts assume-role --role-arn "arn:aws:iam::<accountid>:role/<glue-irc-role>" --role-session-name <glue-irc-role>
```

To access tables through the AWS Glue REST endpoint, you need to initialize a catalog in your Iceberg-compatible client. This initialization requires specifying custom properties, including sigv4 properties, the endpoint URI and the warehouse location. Specify these properties as follows:

- Sigv4 properties - Sigv4 must be enabled, the signing name is `glue`
- Warehouse location - This is your table bucket, specified in this format: *<accountid>*:s3tablescatalog/*<table-bucket-name>*
- Endpoint URI - Refer to the AWS Glue service endpoints reference guide for the region-specific endpoint

The following example shows how to initialize a pyIceberg catalog.

```
rest_catalog = load_catalog(
        s3tablescatalog,
**{
"type": "rest",
"warehouse": "<accountid>:s3tablescatalog/<table-bucket-name>",
"uri": "https://glue.<region>.amazonaws.com/iceberg",
"rest.sigv4-enabled": "true",
"rest.signing-name": "glue",
"rest.signing-region": region
        }
)
```

For additional information about the AWS Glue Iceberg REST endpoint implementation, see Connecting to the Data Catalog using AWS Glue Iceberg REST endpoint in the *AWS Glue User Guide*.

Accessing Amazon S3 tables with an OAuth 2.0 bearer token

External integration partners and third-party query engines can access your Amazon S3 tables using an OAuth 2.0 bearer token instead of SigV4. The client obtains a short-lived, Glue-scoped access token from AWS Sign-In through a token exchange defined in RFC 8693 on the IETF Datatracker website, then presents it to the AWS Glue Iceberg REST endpoint. AWS Glue validates the token and enforces the same Lake Formation permissions described earlier.

Before a client can use this method, complete the identity-federation setup (IAM OIDC identity provider, an IAM role with a web-identity trust policy, and Lake Formation grants). The partner exchanges an end user's JSON Web Token (JWT) for the access token. For the full setup and token exchange details, see Configure OAuth access to AWS Glue Iceberg REST Catalog in the *AWS Sign-In User Guide*.

> **Note**
>
> When a query engine authenticates with an OAuth bearer token, the Lake Formation permissions are enforced on the IAM role that the customer configured for the integration. Grant that role only the permissions the partner's users should have, for example, SELECT and DESCRIBE on specific tables. See Step 3: Grant Lake Formation permissions to the role in the *AWS Sign-In User Guide* (Configure OAuth access to AWS Glue Iceberg REST Catalog).

Accessing tables using the Amazon S3 Tables Iceberg REST endpoint

You can connect your Iceberg REST client to the Amazon S3 Tables Iceberg REST endpoint and make REST API calls to create, update, or query tables in S3 table buckets. The endpoint implements a set of standardized Iceberg REST APIs specified in the Apache Iceberg REST Catalog Open API specification . The endpoint works by translating Iceberg REST API operations into corresponding S3 Tables operations.

> **Note**
>
> Amazon S3 Tables Iceberg REST endpoint can be used to access tables in AWS Partner Network (APN) catalog implementations or custom catalog implementations. It can also be used if you only need basic read/write access to a single table bucket. For other access scenarios we recommend using the AWS Glue Iceberg REST endpoint to connect to tables, which provides unified table management, centralized governance, and fine-grained access

control. For more information, see Accessing Amazon S3 tables using the AWS Glue Iceberg REST endpoint

Configuring the endpoint

You connect to the Amazon S3 Tables Iceberg REST endpoint using the service endpoint. S3 Tables Iceberg REST endpoints have the following format:

```
https://s3tables.<REGION>.amazonaws.com/iceberg
```

Refer to S3 Tables AWS Regions and endpoints for the Region-specific endpoints.

Catalog configuration properties

When using an Iceberg client to connect an analytics engine to the service endpoint, you must specify the following configuration properties when you initialize the catalog. Replace the *placeholder values* with the information for your Region and table bucket.

- The region-specific endpoint as the endpoint URI: `https://s3tables.<REGION>.amazonaws.com/iceberg`
- Your table bucket ARN as the warehouse location: `arn:aws:s3tables:<region>:<accountID>:bucket/<bucketname>`
- Sigv4 properties for authentication. The SigV4 signing name for the service endpoint requests is: `s3tables`

The following examples show you how to configure different clients to use the Amazon S3 Tables Iceberg REST endpoint.

PyIceberg

To use the Amazon S3 Tables Iceberg REST endpoint with PyIceberg, specify the following application configuration properties:

```
rest_catalog = load_catalog(
  catalog_name,
  **{
    "type": "rest",
```

```
    "warehouse":"arn:aws:s3tables:<Region>:<accountID>:bucket/<bucketname>",
    "uri": "https://s3tables.<Region>.amazonaws.com/iceberg",
    "rest.sigv4-enabled": "true",
    "rest.signing-name": "s3tables",
    "rest.signing-region": "<Region>"
  }
)
```

Apache Spark

To use the Amazon S3 Tables Iceberg REST endpoint with Spark, specify the following application configuration properties, replacing the *placeholder values* with the information for your Region and table bucket.

```
spark-shell \
  --packages "org.apache.iceberg:iceberg-spark-
runtime-3.5_2.12:1.4.1,software.amazon.awssdk:bundle:2.20.160,software.amazon.awssdk:url-
connection-client:2.20.160" \
  --master "local[*]" \
  --conf
 "spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions"
 \
  --conf "spark.sql.defaultCatalog=spark_catalog" \
   --conf "spark.sql.catalog.spark_catalog=org.apache.iceberg.spark.SparkCatalog" \
  --conf "spark.sql.catalog.spark_catalog.type=rest" \
  --conf "spark.sql.catalog.spark_catalog.uri=https://
s3tables.<Region>.amazonaws.com/iceberg" \
  --conf
 "spark.sql.catalog.spark_catalog.warehouse=arn:aws:s3tables:<Region>:<accountID>:bucket/
<bucketname>" \
  --conf "spark.sql.catalog.spark_catalog.rest.sigv4-enabled=true" \
  --conf "spark.sql.catalog.spark_catalog.rest.signing-name=s3tables" \
  --conf "spark.sql.catalog.spark_catalog.rest.signing-region=<Region>" \
  --conf "spark.sql.catalog.spark_catalog.io-
impl=org.apache.iceberg.aws.s3.S3FileIO" \
  --conf
 "spark.hadoop.fs.s3a.aws.credentials.provider=org.apache.hadoop.fs.s3a.SimpleAWSCredentialP
 \
  --conf "spark.sql.catalog.spark_catalog.rest-metrics-reporting-enabled=false"
```

Authenticating and authorizing access to the endpoint

API requests to the S3 Tables service endpoints are authenticated using AWS Signature Version 4 (SigV4). See AWS Signature Version 4 for API requests to learn more about AWS SigV4.

The SigV4 signing name for Amazon S3 Tables Iceberg REST endpoint requests is: `s3tables`

Requests to the Amazon S3 Tables Iceberg REST endpoint are authorized using `s3tables` IAM actions corresponding to the REST API operations. These permissions can be defined in either IAM identity-based policies or resource-based policies attached to tables and table buckets. For more information, see Access management for S3 Tables.

You can track requests made to your tables through the REST endpoint with AWS CloudTrail. Requests will be logged as their corresponding S3 IAM action. For example, a `LoadTable` API will generate a management event for the `GetTableMetadataLocation` operation and a data event for the `GetTableData` operation. For more information, see Logging with AWS CloudTrail for S3 Tables.

Prefix and path parameters

Iceberg REST catalog APIs have a free-form prefix in their request URLs. For example, the `ListNamespaces` API call uses the `GET/v1/{prefix}/namespaces` URL format. For S3 Tables the REST path `{prefix}` is always your url-encoded table bucket ARN.

For example, for the following table bucket ARN: `arn:aws:s3tables:`*`us-east-1`*`:111122223333:bucket/`*`bucketname`* the prefix would be: `arn%3Aaws%3As3tables%3A`*`us-east-1`*`%3A`*`111122223333`*`%3Abucket%2F`*`bucketname`*

Namespace path parameter

Namespaces in an Iceberg REST catalog API path can have multiple levels. However, S3 Tables only supports single-level namespaces. To access a namespace in a multi-level catalog hierarchy, you can connect to a multi-level catalog above the namespace when referencing the namespace. This allows any query engine that supports the 3-part notation of `catalog.namespace.table` to access objects in S3 Tables' catalog hierarchy without compatibility issues compared to using the multi-level namespace.

Supported Iceberg REST API operations

The following table contains the supported Iceberg REST APIs and how they correspond to S3 Tables actions.

Iceberg REST operation	REST path	S3 Tables IAM action	CloudTrail EventName
getConfig	GET /v1/config	s3tables:GetTableBucket	s3tables:GetTableBucket
listNamespaces	GET /v1/{prefix}/namespaces	s3tables:ListNamespaces	s3tables:ListNamespaces
createNamespace	POST /v1/{prefix}/namespaces	s3tables:CreateNamespace	s3tables:CreateNamespace
loadNamespaceMetadata	GET /v1/{prefix}/namespaces/{namespace}	s3tables:GetNamespace	s3tables:GetNamespace
dropNamespace	DELETE /v1/{prefix}/namespaces/{namespace}	s3tables:DeleteNamespace	s3tables:DeleteNamespace
listTables	GET /v1/{prefix}/namespaces/{namespace}/tables	s3tables:ListTables	s3tables:ListTables
createTable	POST /v1/{prefix}/namespaces/{namespace}/tables	s3tables:CreateTable ,s3tables:PutTableData	s3tables:CreateTable ,s3tables:PutObject
loadTable	GET /v1/{prefix}/namespaces/{na	s3tables:GetTableMetadataLo	s3tables:GetTableMetadataLo

Iceberg REST operation	REST path	S3 Tables IAM action	CloudTrail EventName
	mespace}/tables/{table}	cation , s3tables:GetTableData	cation , s3tables:GetObject
updateTable	POST /v1/{prefix}/namespaces/{namespace}/tables/{table}	s3tables:UpdateTableMetadataLocation ,s3tables:PutTableData ,s3tables:GetTableData	s3tables:UpdateTableMetadataLocation ,s3tables:PutObject ,s3tables:GetObject
dropTable	DELETE /v1/{prefix}/namespaces/{namespace}/tables/{table}	s3tables:DeleteTable	s3tables:DeleteTable
renameTable	POST /v1/{prefix}/tables/rename	s3tables:RenameTable	s3tables:RenameTable
tableExists	HEAD /v1/{prefix}/namespaces/{namespace}/tables/{table}	s3tables:GetTable	s3tables:GetTable
namespaceExists	HEAD /v1/{prefix}/namespaces/{namespace}	s3tables:GetNamespace	s3tables:GetNamespace

Considerations and limitations

Following are considerations and limitations when using the Amazon S3 Tables Iceberg REST endpoint.

Considerations

- **CreateTable API behavior** – The `stage-create` option is not supported for this operation, and results in a `400 Bad Request` error. This means you cannot create a table from query results using `CREATE TABLE AS SELECT` (CTAS).
- **DeleteTable API behavior** – You can only drop tables with purge enabled. Dropping tables with `purge=false` is not supported and results in a `400 Bad Request` error. Some versions of Spark always set this flag to false even when running `DROP TABLE PURGE` commands. You can try with `DROP TABLE PURGE` or use the S3 Tables DeleteTable operation to delete a table.
- The endpoint only supports standard table metadata operations. For table maintenance, such as snapshot management and compaction, use S3 Tables maintenance API operations. For more information, see S3 Tables maintenance.

Limitations

- Multilevel namespaces are not supported.
- OAuth-based authentication is not supported.
- Only the `owner` property is supported for namespaces.
- View-related APIs defined in the Apache Iceberg REST Open API specification are not supported.
- Running operations on a table with a `metadata.json` file over 50MB is not supported, and will return a `400 Bad Request` error. To control the size of your `metadata.json` files use table maintenance operations. For more information, see S3 Tables maintenance.

Accessing Amazon S3 tables with the Amazon S3 Tables Catalog for Apache Iceberg

You can access S3 tables from open source query engines like Apache Spark by using the Amazon S3 Tables Catalog for Apache Iceberg client catalog. Amazon S3 Tables Catalog for Apache Iceberg is an open source library hosted by AWS Labs. It works by translating Apache Iceberg operations in your query engines (such as table discovery, metadata updates, and adding or removing tables) into S3 Tables API operations.

Amazon S3 Tables Catalog for Apache Iceberg is distributed as a Maven JAR called `s3-tables-catalog-for-iceberg.jar`. You can build the client catalog JAR from the AWS Labs GitHub repository or download it from Maven. When connecting to tables, the client catalog JAR is used as a dependency when you initialize a Spark session for Apache Iceberg.

Using the Amazon S3 Tables Catalog for Apache Iceberg with Apache Spark

You can use the Amazon S3 Tables Catalog for Apache Iceberg client catalog to connect to tables from open-source applications when you initialize a Spark session. In your session configuration you specify Iceberg and Amazon S3 dependencies, and create a custom catalog that uses your table bucket as the metadata warehouse.

Prerequisites

- An IAM identity with access to your table bucket and S3 Tables actions. For more information, see Access management for S3 Tables.

To initialize a Spark session using the Amazon S3 Tables Catalog for Apache Iceberg

- Initialize Spark using the following command. To use the command, replace the replace the Amazon S3 Tables Catalog for Apache Iceberg *`version number`* with the latest version from AWS Labs GitHub repository, and the *`table bucket ARN`* with your own table bucket ARN.

```
spark-shell \
--packages org.apache.iceberg:iceberg-spark-
runtime-3.5_2.12:1.6.1,software.amazon.s3tables:s3-tables-catalog-for-iceberg-
runtime:0.1.4 \
--conf spark.sql.catalog.s3tablesbucket=org.apache.iceberg.spark.SparkCatalog \
--conf spark.sql.catalog.s3tablesbucket.catalog-
impl=software.amazon.s3tables.iceberg.S3TablesCatalog \
--conf spark.sql.catalog.s3tablesbucket.warehouse=arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-table-bucket \
--conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
```

Querying S3 tables with Spark SQL

Using Spark, you can run DQL, DML, and DDL operations on S3 tables. When you query tables you use the fully qualified table name, including the session catalog name following this pattern:

CatalogName.*NamespaceName*.*TableName*

The following example queries show some ways you can interact with S3 tables. To use these example queries in your query engine, replace the *user input placeholder* values with your own.

To query tables with Spark

- Create a namespace

```
spark.sql(" CREATE NAMESPACE IF NOT EXISTS s3tablesbucket.my_namespace")
```

- Create a table

```
spark.sql(" CREATE TABLE IF NOT EXISTS s3tablesbucket.my_namespace.`my_table`
( id INT, name STRING, value INT ) USING iceberg ")
```

- Query a table

```
spark.sql(" SELECT * FROM s3tablesbucket.my_namespace.`my_table` ").show()
```

- Insert data into a table

```
spark.sql(
"""
    INSERT INTO s3tablesbucket.my_namespace.my_table
    VALUES
        (1, 'ABC', 100),
        (2, 'XYZ', 200)
""")
```

- Load an existing data file into a table

1. Read the data into Spark.

```
val data_file_location = "Path such as S3 URI to data file"
val data_file = spark.read.parquet(data_file_location)
```

2. Write the data into an Iceberg table.

```
data_file.writeTo("s3tablesbucket.my_namespace.my_table").using("Iceberg").tableProperty
 ("format-version", "2").createOrReplace()
```

Querying Amazon S3 tables with Athena

Amazon Athena is an interactive query service that you can use to analyze data directly in Amazon S3 by using standard SQL. For more information, see What is Amazon Athena? in the *Amazon Athena User Guide*.

After you integrate your table buckets with AWS analytics services, you can run Data Definition Language (DDL), Data Manipulation Language (DML), and Data Query Language (DQL) queries on S3 tables by using Athena. For more information about how to query tables in a table bucket, see Register S3 Table bucket catalogs in the *Amazon Athena User Guide*.

You can also run queries in Athena from the Amazon S3 console.

Important

When creating tables, make sure that you use all lowercase letters in your table names and table definitions. For example, make sure that your column names are all lowercase. If your table name or table definition contains capital letters, the table isn't supported by AWS Lake Formation or the AWS Glue Data Catalog. In this case, your table won't be visible to AWS analytics services such as Amazon Athena, even if your table buckets are integrated with AWS analytics services.

If your table definition contains capital letters, you receive the following error message when running a SELECT query in Athena: "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names."

Using the S3 console and Amazon Athena

The following procedure uses the Amazon S3 console to access the Athena query editor so that you can query a table with Amazon Athena.

Note

Before performing the following steps, make sure that you've integrated your table buckets with AWS analytics services in this Region. For more information, see the section called "Integrating S3 Tables with AWS analytics services".

To query a table

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. On the **Table buckets** page, choose the bucket that contains the table that you want to query.
4. On the bucket details page, choose the option button next to the name of the table that you want to query.
5. Choose **Query table with Athena**.
6. The Amazon Athena console opens and the Athena query editor appears with a sample SELECT query loaded for you. Modify this query as needed for your use case.

   In the query editor, the **Catalog** field should be populated with **s3tablescatalog/** followed by the name of your table bucket, for example, **s3tablescatalog/*amzn-s3-demo-bucket***. The **Database** field should be populated with the namespace where your table is stored.

   > **Note**
   >
   > If you don't see these values in the **Catalog** and **Database** fields, make sure that you've integrated your table buckets with AWS analytics services in this Region. For more information, see the section called "Integrating S3 Tables with AWS analytics services".

7. To run the query, choose **Run**.

   > **Note**
   >
   > - If you receive the error "Insufficient permissions to execute the query. Principal does not have any privilege on specified resource" when you try to run a query in Athena, you must be granted the necessary Lake Formation permissions on the table. For more information, see the section called "Granting Lake Formation permission on a table or database".
   > - If you receive the error "Iceberg cannot access the requested resource" when you try to run the query, go to the AWS Lake Formation console and make sure that you've granted yourself permissions on the table bucket catalog and database (namespace) that you created. Don't specify a table when granting these permissions. For more

information, see the section called “Granting Lake Formation permission on a table or database”.

- If you receive the following error message when running a SELECT query in Athena, this message is caused by having capital letters in your table name or your column names in your table definition: "GENERIC_INTERNAL_ERROR: Get table request failed: com.amazonaws.services.glue.model.ValidationException: Unsupported Federation Resource - Invalid table or column names." Make sure that your table and column names are all lowercase.

Accessing Amazon S3 tables with Amazon Redshift

Amazon Redshift is a fast, fully managed, petabyte-scale data warehouse service that makes it simple and cost-effective to efficiently analyze all your data using your existing business intelligence tools. Redshift Serverless lets you access and analyze data without all of the configurations of a provisioned data warehouse. For more information, see Get started with serverless data warehouses in the *Amazon Redshift Getting Started Guide*.

Query Amazon S3 tables with Amazon Redshift

Prerequisites

- Integrate your table buckets with AWS analytics services.
  - Create a namespace.
  - Create a table.
- Managing access to a table or database with Lake Formation.

After you complete the prerequisites, you can begin using Amazon Redshift to query tables in one of the following ways:

- Using the Amazon Redshift query editor v2
- Connecting to an Amazon Redshift data warehouse using SQL client tools
- Using the Amazon Redshift Data API

Accessing Amazon S3 tables with Amazon EMR

Amazon EMR (previously called Amazon Elastic MapReduce) is a managed cluster platform that simplifies running big data frameworks, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data. Using these frameworks and related open-source projects, you can process data for analytics purposes and business intelligence workloads. Amazon EMR also lets you transform and move large amounts of data into and out of other AWS data stores and databases.

You can use Apache Iceberg clusters in Amazon EMR to work with S3 tables by connecting to table buckets in a Spark session. To connect to table buckets in Amazon EMR, you can use the AWS analytics services integration through AWS Glue Data Catalog, or you can use the open source Amazon S3 Tables Catalog for Apache Iceberg client catalog.

> **Note**
>
> S3 Tables is supported on Amazon EMR version 7.5 or higher.

Connecting to S3 table buckets with Spark on an Amazon EMR Iceberg cluster

In this procedure, you set up an Amazon EMR cluster configured for Apache Iceberg and then launch a Spark session that connects to your table buckets. You can set this up using the AWS analytics services integration through AWS Glue, or you can use the open source Amazon S3 Tables Catalog for Apache Iceberg client catalog. For information about the client catalog, see Accessing tables using the Amazon S3 Tables Iceberg REST endpoint.

Choose your method of using tables with Amazon EMR from the following options.

Amazon S3 Tables Catalog for Apache Iceberg

The following prerequisites are required to query tables with Spark on Amazon EMR using the Amazon S3 Tables Catalog for Apache Iceberg.

For the latest version of the client catalog JAR, see the s3-tables-catalog GitHub repository.

Prerequisites

- Attach the `AmazonS3TablesFullAccess` policy to the IAM role you use for Amazon EMR.

> **Note**
>
> Apache Iceberg V3 requires Amazon EMR release 7.12 or later. The variant data type requires Amazon EMR release 8.0 or later. The geometry, geography, unknown, and nanosecond-precision timestamp data types require Amazon EMR release 8.1 or later.

To set up an Amazon EMR cluster to query tables with Spark

1. Create a cluster with the following configuration. To use this example, replace the *user input placeholders* with your own information.

```
aws emr create-cluster --release-label emr-7.12.0 \
--applications Name=Spark \
--configurations file://configurations.json \
--region us-east-1 \
--name My_Spark_Iceberg_Cluster \
--log-uri s3://amzn-s3-demo-bucket/ \
--instance-type m5.xlarge \
--instance-count 2 \
--service-role EMR_DefaultRole \
--ec2-attributes \

InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-1234567890abcdef0,KeyName=my-key-pair
```

   `configurations.json`:

```
[{
"Classification":"iceberg-defaults",
"Properties":{"iceberg.enabled":"true"}
}]
```

2. Connect to the Spark primary node using SSH.
3. To initialize a Spark session for Iceberg that connects to your table bucket, enter the following command. Replace the *user input placeholders* with your table bucket ARN.

```
spark-shell \
```

```
--packages software.amazon.s3tables:s3-tables-catalog-for-iceberg-runtime:0.1.8
 \
--conf spark.sql.catalog.s3tablesbucket=org.apache.iceberg.spark.SparkCatalog \
--conf spark.sql.catalog.s3tablesbucket.catalog-
impl=software.amazon.s3tables.iceberg.S3TablesCatalog \
--conf spark.sql.catalog.s3tablesbucket.warehouse=arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-bucket1 \
--conf spark.sql.defaultCatalog=s3tablesbucket \
--conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
```

4. Query your tables with Spark SQL. For example queries, see the section called “Querying S3 tables with Spark SQL”.

AWS analytics services integration

The following prerequisites are required to query tables with Spark on Amazon EMR using the AWS analytics services integration.

Prerequisites

- Integrate your table buckets with AWS analytics services.
- Create the default service role for Amazon EMR (`EMR_DefaultRole_V2`). For details, see Service role for Amazon EMR (EMR role) .
- Create the Amazon EC2 instance profile for Amazon EMR (`EMR_EC2_DefaultRole`). For details, see Service role for cluster EC2 instances (EC2 instance profile).
  - Attach the `AmazonS3TablesFullAccess` policy to `EMR_EC2_DefaultRole`.

To set up an Amazon EMR cluster to query tables with Spark

1. Create a cluster with the following configuration. To use this example, replace the *`user input placeholder`* values with your own information.

> **Note**
>
> Apache Iceberg V3 requires Amazon EMR release 7.12 or later. The variant data type requires Amazon EMR release 8.0 or later. The geometry, geography, unknown, and

nanosecond-precision timestamp data types require Amazon EMR release 8.1 or later.

```
aws emr create-cluster --release-label emr-7.12.0 \
--applications Name=Spark \
--configurations file://configurations.json \
--region us-east-1 \
--name My_Spark_Iceberg_Cluster \
--log-uri s3://amzn-s3-demo-bucket/ \
--instance-type m5.xlarge \
--instance-count 2 \
--service-role EMR_DefaultRole \
--ec2-attributes \

InstanceProfile=EMR_EC2_DefaultRole,SubnetId=subnet-1234567890abcdef0,KeyName=my-key-pair
```

`configurations.json`:

```
[{
"Classification":"iceberg-defaults",
"Properties":{"iceberg.enabled":"true"}
}]
```

2. Connect to the Spark primary node using SSH.
3. Enter the following command to initialize a Spark session for Iceberg that connects to your tables. Replace the *user input placeholders* for Region, account ID and table bucket name with your own information.

```
spark-shell \
--conf
 spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
 \
--conf spark.sql.defaultCatalog=s3tables \
--conf spark.sql.catalog.s3tables=org.apache.iceberg.spark.SparkCatalog \
--conf spark.sql.catalog.s3tables.catalog-
impl=org.apache.iceberg.aws.glue.GlueCatalog \
--conf spark.sql.catalog.s3tables.client.region=us-east-1 \
```

```
--conf spark.sql.catalog.s3tables.glue.id=111122223333:s3tablescatalog/amzn-s3-demo-table-bucket
```

4. Query your tables with Spark SQL. For example queries, see the section called "Querying S3 tables with Spark SQL"

Note

If you are using the `DROP TABLE PURGE` command with Amazon EMR:

- Amazon EMR version 7.5

  Set the Spark config `spark.sql.catalog.your-catalog-name.cache-enabled` to `false`. If this config is set to `true`, run the command in a new session or application so the table cache is not activated.

- Amazon EMR versions higher than 7.5

  `DROP TABLE` is not supported. You can use the S3 Tables `DeleteTable` REST API to delete a table.

Visualizing table data with Quick

Quick is a fast business analytics service to build visualizations, perform ad hoc analysis, and quickly get business insights from your data. Quick seamlessly discovers AWS data sources, enables organizations to scale to hundreds of thousands of users, and delivers fast and responsive query performance by using the Quick Super-fast, Parallel, In-Memory, Calculation Engine (SPICE). For more information, see What is Quick? in the *Quick user guide*.

After you Integrate your table buckets with AWS analytics services, you can create data sets from your tables and work with them in Quick using SPICE or direct SQL queries from your query engine. Quick supports Athena as a data source for S3 tables.

Configure permissions for Quick to access tables

Before working with S3 table data in Quick you must grant permissions to the Quick service role and the Quick admin user permissions on the tables you want to access. Additionally, if you use AWS Lake Formation, you also need to grant Lake Formation permissions to your Quick admin user on those tables you want to access.

Grant permissions to the Quick service role

When set up Quick for the first time in your account, AWS creates a service role that allows Quick to access data sources in other AWS services, such as Athena or Amazon Redshift. The default role name is `aws-quicksight-service-role-v0`.

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Roles** and select the Quick service role. The default name is `aws-quicksight-service-role-v0`
3. Choose **Add permissions** and then **Create inline policy**.
4. Select **JSON** to open the JSON policy editor, then add the following policy.

   JSON

   ```
   {
     "Version":"2012-10-17",
     "Statement": [
       {
         "Sid": "VisualEditor0",
         "Effect": "Allow",
         "Action": "glue:GetCatalog",
         "Resource": "*"
       }
     ]
   }
   ```

5. Choose **Next**, enter a **Policy name** and then **Create policy**.

To configure Lake Formation permissions for the Quick admin user

1. Run the following AWS CLI command to find the ARN of your Quick admin user.

   ```
   aws quicksight list-users --aws-account-id 111122223333 --namespace default --
   region region
   ```

2. Grant Lake Formation permissions to this ARN. For details, see Managing access to a table or database with Lake Formation.

Using table data in Quick

You can connect to table data using Athena as a data source.

Prerequisites

- Integrate your table buckets with AWS analytics services.
  - Create a namespace.
  - Create a table.
  - Configure permissions for Quick to access tables.
- Sign up for Quick.

1. Sign in to your Quick account at https://quicksight.aws.amazon.com/
2. In the dashboard, choose **New analysis**.
3. Choose **New dataset**.
4. Select **Athena**.
5. Enter a **Data source name**, then choose **Create data source**.
6. Choose Use custom SQL. You will not be able to select your table from the **Choose your table** pane.
7. Enter an Athena SQL query that captures the columns you want to visualize, then choose **Confirm query**. For example, use the following query to select all columns:

```
SELECT * FROM "s3tablescatalog/table-bucket-name".namespace.table-name
```

8. Choose **Visualize** to analyze data and start building dashboards. For more information, see Visualizing data in Quick and Exploring interactive dashboards in Quick

Streaming data to tables with Amazon Data Firehose

Amazon Data Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon S3, Amazon Redshift, Amazon OpenSearch Service, Splunk, Apache Iceberg tables, and custom HTTP endpoints or HTTP endpoints owned by supported third-party service providers. With Amazon Data Firehose, you don't need to write applications or manage resources. You configure your data producers to send data to Firehose, and it automatically delivers the data to the destination that you specified. You can also configure Firehose to transform your

data before delivering it. To learn more about Amazon Data Firehose, see What is Amazon Data Firehose?

Complete these steps to set up Firehose streaming to tables in S3 table buckets:

1. Integrate your table buckets with AWS analytics services.
2. Configure Firehose to deliver data into your S3 tables. To do so, you create an AWS Identity and Access Management (IAM) service role that allows Firehose to access your tables.
3. Grant the Firehose service role explicit permissions to your table or table's namespace. For more information, see Grant necessary permissions.
4. Create a Firehose stream that routes data to your table.

Creating a role for Firehose to use S3 tables as a destination

Firehose needs an IAM service role with specific permissions to access AWS Glue tables and write data to S3 tables. You need this provide this IAM role when you create a Firehose stream.

1. Open the IAM console at https://console.aws.amazon.com/iam/.
2. In the left navigation pane, choose **Policies**
3. Choose **Create a policy**, and choose **JSON** in policy editor.
4. Add the following inline policy that grants permissions to all databases and tables in your data catalog. If you want, you can give permissions only to specific tables and databases. To use this policy, replace the *`user input placeholders`* with your own information.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "S3TableAccessViaGlueFederation",
            "Effect": "Allow",
            "Action": [
                "glue:GetTable",
                "glue:GetDatabase",
                "glue:UpdateTable"
            ],
            "Resource": [
```

```
                "arn:aws:glue:us-east-1:111122223333:catalog/s3tablescatalog/
*",
                "arn:aws:glue:us-east-1:111122223333:catalog/
s3tablescatalog",
                "arn:aws:glue:us-east-1:111122223333:catalog",
                "arn:aws:glue:us-east-1:111122223333:database/*",
                "arn:aws:glue:us-east-1:111122223333:table/*/*"
            ]
        },
        {
            "Sid": "S3DeliveryErrorBucketPermission",
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:GetBucketLocation",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:ListBucketMultipartUploads",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::error delivery bucket",
                "arn:aws:s3:::error delivery bucket/*"
            ]
        },
        {
            "Sid": "RequiredWhenUsingKinesisDataStreamsAsSource",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetShardIterator",
                "kinesis:GetRecords",
                "kinesis:ListShards"
            ],
            "Resource": "arn:aws:kinesis:us-
east-1:111122223333:stream/stream-name"
        },
        {
            "Sid":
 "RequiredWhenDoingMetadataReadsANDDataAndMetadataWriteViaLakeformation",
            "Effect": "Allow",
            "Action": [
                "lakeformation:GetDataAccess"
            ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "RequiredWhenUsingKMSEncryptionForS3ErrorBucketDelivery",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:us-east-1:111122223333:key/KMS-key-id"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.us-east-1.amazonaws.com"
                },
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3:::error
 delivery bucket/prefix*"
                }
            }
        },
        {
            "Sid": "LoggingInCloudWatch",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:111122223333:log-group:log-group-
name:log-stream:log-stream-name"
            ]
        },
        {
            "Sid": "RequiredWhenAttachingLambdaToFirehose",
            "Effect": "Allow",
            "Action": [
                "lambda:InvokeFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Resource": [
                "arn:aws:lambda:us-east-1:111122223333:function:function-
name:function-version"
            ]
```

```
        }
    ]
}
```

This policy has a statements that allow access to Kinesis Data Streams, invoking Lambda functions and access to AWS KMS keys. If you don't use any of these resources, you can remove the respective statements.

If error logging is enabled, Firehose also sends data delivery errors to your CloudWatch log group and streams. For this, you must configure log group and log stream names. For log group and log stream names, see Monitor Amazon Data Firehose Using CloudWatch Logs.

5. After you create the policy, create an IAM role with **AWS service** as the **Trusted entity type**.
6. For **Service or use case**, choose **Kinesis**. For **Use case** choose **Kinesis Firehose**.
7. Choose **Next**, and then select the policy you created earlier.
8. Give your role a name. Review your role details, and choose **Create role**. The role will have the following trust policy.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "firehose.amazonaws.com"
                ]
            }
        }
    ]
}
```

Creating a Firehose stream to S3 tables

The following procedure shows how to create a Firehose stream to deliver data to S3 tables using the console. The following prerequisites are required to set up a Firehose stream to S3 tables.

Prerequisites

- Integrate your table buckets with AWS analytics services.
  - Create a namespace.
  - Create a table.
- Create the Role for Firehose to access S3 Tables.
- Grant necessary permissions to the Firehose service role you created to access tables.

To provide routing information to Firehose when you configure a stream, you use your namespace as the database name and the name of a table in that namespace. You can use these values in the Unique key section of a Firehose stream configuration to route data to a single table. You can also use this values to route to a table using JSON Query expressions. For more information, see Route incoming records to a single Iceberg table.

To set up a Firehose stream to S3 tables (Console)

1. Open the Firehose console at https://console.aws.amazon.com/firehose/.
2. Choose **Create Firehose stream**.
3. For **Source**, choose one of the following sources:
   - Amazon Kinesis Data Streams
   - Amazon MSK
   - Direct PUT
4. For **Destination**, choose **Apache Iceberg Tables**.
5. Enter a **Firehose stream name**.
6. Configure your **Source settings**.
7. For **Destination settings**, choose **Current account** to stream to tables in your account or **Cross-account** for tables in another account.
   - For tables in the **Current account**, select your S3 Tables catalog from the **Catalog** dropdown.

- For tables in a **Cross-account**, enter the **Catalog ARN** of the catalog you want to stream to in another account.

8. Configure database and table names using **Unique Key configuration**, JSONQuery expressions, or in a Lambda function. For more information, refer to Route incoming records to a single Iceberg table and Route incoming records to different Iceberg tables in the *Amazon Data Firehose Developer Guide*.
9. Under **Backup settings**, specify a **S3 backup bucket**.
10. For **Existing IAM roles** under **Advanced settings**, select the IAM role you created for Firehose.
11. Choose **Create Firehose stream**.

For more information about the other settings that you can configure for a stream, see Set up the Firehose stream in the *Amazon Data Firehose Developer Guide*.

Running ETL jobs on Amazon S3 tables with AWS Glue

AWS Glue is a serverless data integration service that makes it easy for analytics users to discover, prepare, move, and integrate data from multiple sources. You can use AWS Glue jobs to run extract, transform, and load (ETL) pipelines to load data into your data lakes. For more information about AWS Glue, see What is AWS Glue? in the *AWS Glue Developer Guide*.

An AWS Glue job encapsulates a script that connects to your source data, processes it, and then writes it out to your data target. Typically, a job runs extract, transform, and load (ETL) scripts. Jobs can run scripts designed for Apache Spark runtime environments. You can monitor job runs to understand runtime metrics such as completion status, duration, and start time.

You can use AWS Glue jobs to process data in your S3 tables by connecting to your tables through the integration with AWS analytics services, or, connect directly using the Amazon S3 Tables Iceberg REST endpoint or the Amazon S3 Tables Catalog for Apache Iceberg. This guide covers the basic steps to get started using AWS Glue with S3 Tables, including:

Topics

- Step 1 – Prerequisites
- Step 2 – Create a script to connect to table buckets
- Step 3 – Create a AWS Glue job that queries tables

Choose your access method based on your specific AWS Glue ETL job requirements:

- **AWS analytics services integration (Recommended)** – Recommended when you need centralized metadata management across multiple AWS analytics services, need to leverage existing AWS Glue Data Catalog permissions and optionally Lake Formation, or are building production ETL pipelines that integrate with other AWS services like Athena or Amazon EMR.
- **Amazon S3 Tables Iceberg REST endpoint** – Recommended when you need to connect to S3 tables from third-party query engines that support Apache Iceberg, build custom ETL applications that need direct REST API access, or when you require control over catalog operations without dependencies on AWS Glue Data Catalog.
- **Amazon S3 Tables Catalog for Apache Iceberg** – Use only for legacy applications or specific programmatic scenarios that require the Java client library. This method is not recommended for new AWS Glue ETL job implementations due to additional `JAR` dependency management and complexity.

> **Note**
>
> S3 Tables is supported on AWS Glue version 5.0 or higher. Apache Iceberg V3 requires AWS Glue version 5.1 or higher, and the V3 data types require AWS Glue version 6.0 or higher.

Step 1 – Prerequisites

Before you can query tables from a AWS Glue job you must configure an IAM role that AWS Glue can use to run the job. Choose your method of access to see specific prerequisites for that method.

AWS analytics services integration (Recommended)

Prerequisites required to use the S3 Tables AWS analytics integration to run AWS Glue jobs.

- Integrate your table buckets with AWS analytics services.
- Create an IAM role for AWS Glue.
  - Attach the `AmazonS3TablesFullAccess` managed policy to the role.
  - Attach the `AmazonS3FullAccess` managed policy to the role.

Amazon S3 Tables Iceberg REST endpoint

Prerequisites to use the Amazon S3 Tables Iceberg REST endpoint to run AWS Glue ETL jobs.

- Create an IAM role for AWS Glue.
  - Attach the `AmazonS3TablesFullAccess` managed policy to the role.
  - Attach the `AmazonS3FullAccess` managed policy to the role.

Amazon S3 Tables Catalog for Apache Iceberg

Prerequisites use the Amazon S3 Tables Catalog for Apache Iceberg to run AWS Glue ETL jobs.

- Create an IAM role for AWS Glue.
  - Attach the `AmazonS3TablesFullAccess` managed policy to the role.
  - Attach the `AmazonS3FullAccess` managed policy to the role.
  - To use the Amazon S3 Tables Catalog for Apache Iceberg you need to download the client catalog JAR and upload it to an S3 bucket.

    Downloading the catalog JAR

    1. Check for the latest version on Maven Central. You can download the JAR from Maven central using your browser, or using the following command. Make sure to replace the *`version number`* with the latest version.

       ```
       wget https://repo1.maven.org/maven2/software/amazon/s3tables/s3-tables-catalog-for-iceberg-runtime/0.1.5/s3-tables-catalog-for-iceberg-runtime-0.1.5.jar
       ```

    2. Upload the downloaded JAR to an S3 bucket that your AWS Glue IAM role can access. You can use the following AWS CLI command to upload the JAR. Make sure to replace the *`version number`* with the latest version, and the *`bucket name`* and *`path`* with your own.

       ```
       aws s3 cp s3-tables-catalog-for-iceberg-runtime-0.1.5.jar s3://amzn-s3-demo-bucket/jars/
       ```

Step 2 – Create a script to connect to table buckets

To access your table data when you run an AWS Glue ETL job, you configure a Spark session for Apache Iceberg that connects to your S3 table bucket. You can modify an existing script to connect

to your table bucket or create a new script. For more information on creating AWS Glue scripts, see Tutorial: Writing an AWS Glue for Spark script in the *AWS Glue Developer Guide*.

You can configure the session to connect to your table buckets through the any of the following S3 Tables access methods:

- S3 Tables AWS analytics services integration (Recommended)
- Amazon S3 Tables Iceberg REST endpoint
- Amazon S3 Tables Catalog for Apache Iceberg

Choose from the following access methods to view setup instructions and configuration examples.

AWS analytics services integration (Recommended)

As a prerequisites to query tables with Spark on AWS Glue using the AWS analytics services integration, you must Integrate your table buckets with AWS analytics services

You can configure the connection to your table bucket through a Spark session in a job or with AWS Glue Studio magics in an interactive session. To use the following examples, replace the *placeholder values* with the information for your own table bucket.

Using a PySpark script

Use the following code snippet in a PySpark script to configure a AWS Glue job to connect to your table bucket using the integration.

```
spark = SparkSession.builder.appName("SparkIcebergSQL") \
    .config("spark.jars.packages", "org.apache.iceberg:iceberg-spark-
runtime-3.4_2.12:1.4.2") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog","s3tables") \
    .config("spark.sql.catalog.s3tables",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3tables.catalog-impl",
 "org.apache.iceberg.aws.glue.GlueCatalog") \
    .config("spark.sql.catalog.s3tables.glue.id",
 "111122223333:s3tablescatalog/amzn-s3-demo-table-bucket") \
    .config("spark.sql.catalog.s3tables.warehouse", "s3://amzn-s3-demo-table-
bucket/warehouse/") \
```

```
    .getOrCreate()
```

Using an interactive AWS Glue session

If you are using an interactive notebook session with AWS Glue 5.0, specify the same configurations using the `%%configure` magic in a cell prior to code execution.

```
%%configure
{"conf":
 "spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
 --conf spark.sql.defaultCatalog=s3tables --conf
 spark.sql.catalog.s3tables=org.apache.iceberg.spark.SparkCatalog --conf
 spark.sql.catalog.s3tables.catalog-impl=org.apache.iceberg.aws.glue.GlueCatalog
 --conf spark.sql.catalog.s3tables.glue.id=111122223333:s3tablescatalog/amzn-s3-
demo-table-bucket --conf spark.sql.catalog.s3tables.warehouse=s3://amzn-s3-demo-
table-bucket/warehouse/"}
```

Amazon S3 Tables Iceberg REST endpoint

You can configure the connection to your table bucket through a Spark session in a job or with AWS Glue Studio magics in an interactive session. To use the following examples, replace the *placeholder values* with the information for your own table bucket.

Using a PySpark script

Use the following code snippet in a PySpark script to configure a AWS Glue job to connect to your table bucket using the endpoint.

```
spark = SparkSession.builder.appName("glue-s3-tables-rest") \
    .config("spark.jars.packages", "org.apache.iceberg:iceberg-spark-
runtime-3.4_2.12:1.4.2") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog", "s3_rest_catalog") \
    .config("spark.sql.catalog.s3_rest_catalog",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3_rest_catalog.type", "rest") \
    .config("spark.sql.catalog.s3_rest_catalog.uri", "https://
s3tables.Region.amazonaws.com/iceberg") \
    .config("spark.sql.catalog.s3_rest_catalog.warehouse",
 "arn:aws:s3tables:Region:111122223333:bucket/amzn-s3-demo-table-bucket") \
```

```
    .config("spark.sql.catalog.s3_rest_catalog.rest.sigv4-enabled", "true") \
    .config("spark.sql.catalog.s3_rest_catalog.rest.signing-name", "s3tables") \
    .config("spark.sql.catalog.s3_rest_catalog.rest.signing-region", "Region") \
    .config('spark.sql.catalog.s3_rest_catalog.io-
impl','org.apache.iceberg.aws.s3.S3FileIO') \
    .config('spark.sql.catalog.s3_rest_catalog.rest-metrics-reporting-
enabled','false') \
    .getOrCreate()
```

Using an interactive AWS Glue session

If you are using an interactive notebook session with AWS Glue 5.0, specify the same configurations using the `%%configure` magic in a cell prior to code execution. Replace the placeholder values with the information for your own table bucket.

```
%%configure
{"conf":
 "spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
 --conf spark.sql.defaultCatalog=s3_rest_catalog --conf
 spark.sql.catalog.s3_rest_catalog=org.apache.iceberg.spark.SparkCatalog
 --conf spark.sql.catalog.s3_rest_catalog.type=rest --conf
 spark.sql.catalog.s3_rest_catalog.uri=https://s3tables.Region.amazonaws.com/
iceberg --conf
 spark.sql.catalog.s3_rest_catalog.warehouse=arn:aws:s3tables:Region:111122223333:bucket/
amzn-s3-demo-table-bucket --conf spark.sql.catalog.s3_rest_catalog.rest.sigv4-
enabled=true --conf spark.sql.catalog.s3_rest_catalog.rest.signing-name=s3tables
 --conf spark.sql.catalog.s3_rest_catalog.rest.signing-region=Region --conf
 spark.sql.catalog.s3_rest_catalog.io-impl=org.apache.iceberg.aws.s3.S3FileIO --
conf spark.sql.catalog.s3_rest_catalog.rest-metrics-reporting-enabled=false"}
```

Amazon S3 Tables Catalog for Apache Iceberg

As a prerequisite to connecting to tables using the Amazon S3 Tables Catalog for Apache Iceberg you must first download the latest catalog jar and upload it to an S3 bucket. Then, when you create your job, you add the the path to the client catalog JAR as a special parameter. For more information on job parameters in AWS Glue, see Special parameters used in AWS Glue jobs in the *AWS Glue Developer Guide*.

You can configure the connection to your table bucket through a Spark session in a job or with AWS Glue Studio magics in an interactive session. To use the following examples, replace the *`placeholder values`* with the information for your own table bucket.

Using a PySpark script

Use the following code snippet in a PySpark script to configure a AWS Glue job to connect to your table bucket using the JAR. Replace the placeholder values with the information for your own table bucket.

```
spark = SparkSession.builder.appName("glue-s3-tables") \
    .config("spark.jars.packages", "org.apache.iceberg:iceberg-spark-
runtime-3.4_2.12:1.4.2") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog", "s3tablesbucket") \
    .config("spark.sql.catalog.s3tablesbucket",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3tablesbucket.catalog-impl",
 "software.amazon.s3tables.iceberg.S3TablesCatalog") \
    .config("spark.sql.catalog.s3tablesbucket.warehouse",
 "arn:aws:s3tables:Region:111122223333:bucket/amzn-s3-demo-table-bucket") \
    .getOrCreate()
```

Using an interactive AWS Glue session

If you are using an interactive notebook session with AWS Glue 5.0, specify the same configurations using the `%%configure` magic in a cell prior to code execution. Replace the placeholder values with the information for your own table bucket.

```
%%configure
{"conf":
 "spark.sql.extensions=org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions
 --conf spark.sql.defaultCatalog=s3tablesbucket --conf
 spark.sql.catalog.s3tablesbucket=org.apache.iceberg.spark.SparkCatalog
 --conf spark.sql.catalog.s3tablesbucket.catalog-
impl=software.amazon.s3tables.iceberg.S3TablesCatalog --conf
 spark.sql.catalog.s3tablesbucket.warehouse=arn:aws:s3tables:Region:111122223333:bucket/
amzn-s3-demo-table-bucket", "extra-jars": "s3://amzn-s3-demo-bucket/jars/s3-
tables-catalog-for-iceberg-runtime-0.1.5.jar"}
```

Sample scripts

The following example PySpark scripts can be used to test querying S3 tables with an AWS Glue job. These scripts connect to your table bucket and runs queries to: create a new namespace, create

a sample table, insert data into the table, and return the table data. To use the scripts, replace the *placeholder values* with the information for you own table bucket.

Choose from the following scripts based on your S3 Tables access method.

S3 Tables integration with AWS analytics services

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("SparkIcebergSQL") \
    .config("spark.jars.packages", "org.apache.iceberg:iceberg-spark-
runtime-3.4_2.12:1.4.2") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog","s3tables") \
    .config("spark.sql.catalog.s3tables", "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3tables.catalog-impl",
 "org.apache.iceberg.aws.glue.GlueCatalog") \
    .config("spark.sql.catalog.s3tables.glue.id",
 "111122223333:s3tablescatalog/amzn-s3-demo-table-bucket") \
    .config("spark.sql.catalog.s3tables.warehouse", "s3://amzn-s3-demo-table-bucket/
bucket/amzn-s3-demo-table-bucket") \
    .getOrCreate()

namespace = "new_namespace"
table = "new_table"

spark.sql("SHOW DATABASES").show()

spark.sql(f"DESCRIBE NAMESPACE {namespace}").show()

spark.sql(f"""
    CREATE TABLE IF NOT EXISTS {namespace}.{table} (
       id INT,
       name STRING,
       value INT
    )
""")

spark.sql(f"""
    INSERT INTO {namespace}.{table}
    VALUES
       (1, 'ABC', 100),
       (2, 'XYZ', 200)
```

```
""")

spark.sql(f"SELECT * FROM {namespace}.{table} LIMIT 10").show()
```

Amazon S3 Tables Iceberg REST endpoint

```
from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("glue-s3-tables-rest") \
    .config("spark.jars.packages", "org.apache.iceberg:iceberg-spark-
runtime-3.4_2.12:1.4.2") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog", "s3_rest_catalog") \
    .config("spark.sql.catalog.s3_rest_catalog",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3_rest_catalog.type", "rest") \
    .config("spark.sql.catalog.s3_rest_catalog.uri", "https://
s3tables.Region.amazonaws.com/iceberg") \
    .config("spark.sql.catalog.s3_rest_catalog.warehouse",
 "arn:aws:s3tables:Region:111122223333:bucket/amzn-s3-demo-table-bucket") \
    .config("spark.sql.catalog.s3_rest_catalog.rest.sigv4-enabled", "true") \
    .config("spark.sql.catalog.s3_rest_catalog.rest.signing-name", "s3tables") \
    .config("spark.sql.catalog.s3_rest_catalog.rest.signing-region", "Region") \
    .config('spark.sql.catalog.s3_rest_catalog.io-
impl','org.apache.iceberg.aws.s3.S3FileIO') \
    .config('spark.sql.catalog.s3_rest_catalog.rest-metrics-reporting-
enabled','false') \
    .getOrCreate()

namespace = "s3_tables_rest_namespace"
table = "new_table_s3_rest"

spark.sql("SHOW DATABASES").show()

spark.sql(f"DESCRIBE NAMESPACE {namespace}").show()

spark.sql(f"""
    CREATE TABLE IF NOT EXISTS {namespace}.{table} (
       id INT,
       name STRING,
       value INT
    )
```

```
""")

spark.sql(f"""
    INSERT INTO {namespace}.{table}
    VALUES
       (1, 'ABC', 100),
       (2, 'XYZ', 200)
""")

spark.sql(f"SELECT * FROM {namespace}.{table} LIMIT 10").show()
```

Amazon S3 Tables Catalog for Apache Iceberg

```
from pyspark.sql import SparkSession

#Spark session configurations
spark = SparkSession.builder.appName("glue-s3-tables") \
    .config("spark.jars.packages", "org.apache.iceberg:iceberg-spark-
runtime-3.4_2.12:1.4.2") \
    .config("spark.sql.extensions",
 "org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensions") \
    .config("spark.sql.defaultCatalog", "s3tablesbucket") \
    .config("spark.sql.catalog.s3tablesbucket",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3tablesbucket.catalog-impl",
 "software.amazon.s3tables.iceberg.S3TablesCatalog") \
    .config("spark.sql.catalog.s3tablesbucket.warehouse",
 "arn:aws:s3tables:Region:111122223333:bucket/amzn-s3-demo-table-bucket") \
    .getOrCreate()

#Script
namespace = "new_namespace"
table = "new_table"

spark.sql(f"CREATE NAMESPACE IF NOT EXISTS s3tablesbucket.{namespace}")
spark.sql(f"DESCRIBE NAMESPACE {namespace}").show()

spark.sql(f"""
    CREATE TABLE IF NOT EXISTS {namespace}.{table} (
       id INT,
       name STRING,
```

```
        value INT
    )
""")

spark.sql(f"""
    INSERT INTO {namespace}.{table}
    VALUES
        (1, 'ABC', 100),
        (2, 'XYZ', 200)
""")

spark.sql(f"SELECT * FROM {namespace}.{table} LIMIT 10").show()
```

Step 3 – Create a AWS Glue job that queries tables

The following procedures show how to setup AWS Glue jobs that connect to your S3 table buckets. You can do this using the AWS CLI or using the console with AWS Glue Studio script editor. For more information, see Authoring jobs in AWS Glue in the *AWS Glue User Guide*.

Using AWS Glue Studio script editor

The following procedure shows how to use the AWS Glue Studio script editor to create an ETL job that queries your S3 tables.

Prerequisites

- Step 1 – Prerequisites
- Step 2 – Create a script to connect to table buckets

1. Open the AWS Glue console at https://console.aws.amazon.com/glue/.
2. From the Navigation pane, choose **ETL jobs**.
3. Choose **Script editor**, then choose **Upload script** and upload the PySpark script you created to query S3 tables.
4. Select the **Job details** tab and enter the following for **Basic properties**.
   - For **Name**, enter a name for the job.
   - For **IAM Role**, select the role you created for AWS Glue.
5. (Optional) If you are using the Amazon S3 Tables Catalog for Apache Iceberg access method, expand **Advanced properties** and for **Dependent JARs path**, enter the S3 URI of the client

catalog jar your uploaded to an S3 bucket as a prerequisite. For example, s3://*amzn-s3-demo-bucket1*/*jars*/s3-tables-catalog-for-iceberg-runtime-*0.1.5*.jar

6. Choose **Save** to create the job.
7. Choose **Run** start the job, and review the job status under the **Runs** tab.

Using the AWS CLI

The following procedure shows how to use the AWS CLI to create an ETL job that queries your S3 tables. To use the commands replace the *placeholder values* with your own.

Prerequisites

- Step 1 – Prerequisites
- Step 2 – Create a script to connect to table buckets and upload it to an S3 bucket.

1. Create an AWS Glue job.

```
aws glue create-job \
--name etl-tables-job \
--role arn:aws:iam::111122223333:role/AWSGlueServiceRole \
--command '{
    "Name": "glueetl",
    "ScriptLocation": "s3://amzn-s3-demo-bucket1/scripts/glue-etl-query.py",
    "PythonVersion": "3"
}' \
--default-arguments '{
    "--job-language": "python",
    "--class": "GlueApp"
}' \
--glue-version "5.0"
```

> **Note**
>
> (Optional) If you are using the Amazon S3 Tables Catalog for Apache Iceberg access method, add the client catalog JAR to the `--default-arguments` using the `--extra-jars` parameter. Replace the *input placeholders* with your own when you add the parameter.

```
                                "--extra-jars": "s3://amzn-s3-demo-bucket/jar-
path/s3-tables-catalog-for-iceberg-runtime-0.1.5.jar"
```

2. Start your job.

```
aws glue start-job-run \
--job-name etl-tables-job
```

3. To review you job status, copy the run ID from the previous command and enter it into the following command.

```
aws glue get-job-run --job-name etl-tables-job \
--run-id jr_ec9a8a302e71f8483060f87b6c309601ea9ee9c1ffc2db56706dfcceb3d0e1ad
```

Getting started querying S3 Tables with Amazon SageMaker Unified Studio

Amazon SageMaker Unified Studio is a comprehensive analytics service that enables you to query and derive insights from your data using SQL, natural language, and interactive notebooks. It supports team collaboration and analysis workflows across AWS data repositories and third-party sources within a unified interface. SageMaker Unified Studio integrates directly with S3 Tables, providing a seamless transition from data storage to analysis within the Amazon S3 console.

You can integrate S3 Tables with SageMaker Unified Studio through the Amazon S3 console or SageMaker Unified Studio console.

For setup through the SageMaker Unified Studio console, see the SageMaker Unified Studio documentation.

Requirements for querying S3 Tables with SageMaker Unified Studio

Using SageMaker Unified Studio with S3 Tables requires the following:

- Your table buckets have been integrated with AWS analytics services in the current Region. For more information, see Integrating S3 Tables with AWS analytics services.
- You are using an IAM role with permissions to create and view resources in SageMaker Unified Studio. For more information, see Setup IAM-based domains in SageMaker Unified Studio.

- You have a SageMaker domain and project. For more information, see Domains in the *SageMaker Unified Studio Administrator Guide*, and Projects in the *SageMaker Unified Studio User Guide*.

If you haven't already performed these actions or created these resources, S3 Tables can automatically complete this set up for you so you can begin querying with SageMaker Unified Studio.

Getting started querying S3 Tables with SageMaker Unified Studio

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. On the **Table buckets** page, choose the bucket that contains the table that you want to query.
4. On the bucket details page, select the table that you want to query.
5. Choose **Query**.
6. Then, choose **Query table in SageMaker Unified Studio**.

    a. If you've already configured SageMaker Unified Studio for your tables the SageMaker Unified Studio console opens to the query editor with a sample SELECT query loaded for you. Modify this query as needed for your use case.

    b. If you haven't already configured SageMaker Unified Studio for S3 Tables, a set up page appears with a single step to enable Integration with AWS analytics services which integrates your tables with services like SageMaker Unified Studio. This step will execute automatically then you will be redirected to a page in the SageMaker Unified Studio console with the following options to configure your account for querying S3 Tables:

        i. In **Setting you up as an administrator**, your current federated IAM role is selected. If your current role does not already have the required permissions, you will need to setup an IAM-based domain in SageMaker Unified Studio and assign permissions to your role so you can login to SageMaker Unified Studio.

        ii. In **Project data and administrative control**, select **Auto-create a new role with required permissions** to automatically create a role with the required permissions, or select **Use an existing role** and choose a role. If the chosen role does not already have the required permissions, you will need to setup an IAM-based domain in SageMaker Unified Studio and assign permissions to your admin execution role so you can access data in SageMaker Unified Studio.

iii. In **Data encryption** select **Use AWS owned key** to let AWS own and manage a key for you or **Choose a different AWS AWS KMS key (advanced)** to use an existing key or to create a new one.

iv. Select **Set up SageMaker Unified Studio**.

v. Next, the SageMaker Unified Studio console opens to the query editor with a sample `SELECT` query loaded for you. Modify this query as needed for your use case.

In the query editor, the **Catalog** field should be populated with `s3tablescatalog/` followed by the name of your table bucket, for example, `s3tablescatalog/amzn-s3-demo-table-bucket`. The **Database** field is populated with the namespace where your table is stored.

Streaming tables with Amazon MSK

Streaming tables are Apache Iceberg tables in your table buckets that Amazon Managed Streaming for Apache Kafka (Amazon MSK) writes to continuously. Unlike tables that you create and write to directly, a streaming table is created and populated by Amazon MSK. Amazon MSK Express brokers deliver records from a Kafka topic as Apache Parquet data files and commit them to the table, making the data available to query within minutes of being produced. A single streaming table scales to Amazon MSK delivery throughput of up to 10 GBps without manual scaling.

Because Amazon MSK owns the writes, a streaming table is read-only. You have full read access from any Iceberg-compatible engine, and you control table maintenance and read access the same way as for other tables in your table bucket.

You create a streaming table by configuring an Amazon MSK delivery channel rather than by calling `CreateTable`. Amazon MSK creates the table on first delivery. For how to configure the channel, including the source topic, data freshness, dead-letter queue, and the AWS Glue Schema Registry, see Amazon MSK data delivery in the *Amazon Managed Streaming for Apache Kafka Developer Guide*.

What you can and can't do with streaming tables

With a streaming table, you can do the following:

- Query the table with Amazon Athena, Amazon Redshift, Amazon EMR, Apache Spark, or any other Iceberg-compatible query engine.

- Grant read access with AWS Identity and Access Management (IAM) policies, table bucket policies, and AWS Lake Formation.
- Configure the automated S3 Tables maintenance jobs: compaction, snapshot expiration, and unreferenced file removal.
- Configure record expiration on the `timestamptz` partition column to automatically delete records older than a retention period that you specify.
- Delete the table with `DeleteTable`. After you delete the table, Amazon MSK delivery fails. To resume delivery, recreate the Amazon MSK channel.

With a streaming table, you can't do the following:

- Write to the table directly (`INSERT`, `UPDATE`, `MERGE`, `COPY INTO`, or Iceberg writes from your own engines).
- Change the schema, partition spec, sort order, or table properties.
- Delete individual rows outside of the record expiration job.
- Backfill records that were produced to the Kafka topic before the Amazon MSK channel was enabled.

Permissions to query streaming tables

Read access to a streaming table works the same as for any other Amazon S3 table. You grant it with IAM policies, table bucket policies, and AWS Lake Formation. The following example policy allows you to query a streaming table. If the table is encrypted with an AWS AWS KMS key, the second statement grants the `kms:Decrypt` permission on that key.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"PermissionsToQueryStreamingTables",
         "Effect":"Allow",
         "Action":[
             "s3tables:GetTable",
             "s3tables:GetTableData",
             "s3tables:GetTableMetadataLocation"
         ],
         "Resource":[
```

```
            "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket",
            "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket/
table/*"
         ]
      },
      {
         "Sid":"PermissionsToDecryptTableData",
         "Effect":"Allow",
         "Action":[
             "kms:Decrypt"
         ],
         "Resource":[
            "arn:aws:kms:us-east-1:111122223333:key/01234567-89ab-
cdef-0123-456789abcdef"
         ]
       }
    ]
}
```

The permissions that Amazon MSK needs to create the table and write to it belong to the Amazon MSK service role. For those permissions, see IAM permissions for Amazon MSK data delivery in the *Amazon Managed Streaming for Apache Kafka Developer Guide*. If your table bucket is in a different AWS account from the Amazon MSK cluster, the bucket owner must attach a table bucket policy that grants the Amazon MSK service role the actions it needs. For more information, see Managing table bucket policies.

Working with Apache Iceberg V3

Apache Iceberg Version 3 (V3) is the latest version of the Apache Iceberg table format specification, introducing advanced capabilities for building petabyte-scale data lakes with improved performance and reduced operational overhead. V3 addresses common performance bottlenecks encountered with Version 2 (V2), particularly around batch updates and compliance deletes.

AWS provides support for the capabilities defined in the Apache Iceberg Version 3 (V3) specification. These capabilities include deletion vectors, row lineage, and column default values. They also include the following data types:

- variant
- geometry
- geography

- unknown
- nanosecond-precision timestamps without a time zone
- nanosecond-precision timestamps with a time zone

S3 Tables support all of the data types that V3 introduces.

You can use Apache Iceberg V3 with Apache Spark on Amazon EMR, AWS Glue ETL, Amazon SageMaker Unified Studio Notebooks, and tables in AWS Glue Data Catalog, including Amazon S3 Tables.

Key Features in V3

Deletion Vectors

Replaces V2's positional delete files with an efficient binary format stored as Puffin files. This eliminates write amplification from random batch updates and GDPR compliance deletes, significantly reducing the overhead of maintaining fresh data. Organizations processing high-frequency updates will see immediate improvements in write performance and reduced storage costs from fewer small files.

Row-lineage

Enables precise change tracking at the row level. Your downstream systems can process changes incrementally, speeding up data pipelines and reducing compute costs for change data capture (CDC) workflows. This built-in capability eliminates the need for custom change tracking implementations.

Variant data type

With the variant data type, you can write semi-structured data like JSON directly in Iceberg tables without defining a fixed schema in advance. V3 compatible engines shred your semi-structured data into hidden columns as you write it, generating Parquet column statistics that query engines use for optimizations like file pruning. This reduces the data your analytical queries scan. S3 Tables provides ongoing table maintenance for variant columns, including compaction, so you can consolidate data from semi-structured sources into larger files that Iceberg engines can read efficiently.

Geospatial data types

The `geometry` and `geography` types store points, lines, and polygons as native columns instead of encoded strings or paired latitude and longitude values. `geometry` operates on a

Cartesian plane. `geography` treats coordinates as spherical coordinates on a spheroid. Both types recognize spatial reference system identifier (SRID) 0 and SRID 4326. Workloads such as fleet tracking and asset mapping can filter on location at query time. For examples, see the section called “Using the geospatial data types”.

Nanosecond-precision timestamps

`timestamp(9)` records event times to nanosecond precision without a time zone, and `timestamptz(9)` records them with a time zone. Telemetry, sensor fusion, and financial workloads can store timestamps at source precision instead of encoding them as integers and converting them on read. For examples, see the section called “Using nanosecond-precision timestamps”.

Column default values

A field can carry an `initial-default` value, used to populate the field for records written before the field was added to the schema, and a `write-default` value, used when a writer omits the column. Column default values populate a newly added column for existing rows, with no backfill. For examples, see the section called “Using column default values”.

> **Important**
>
> The V3 specification requires that all columns of unknown, `variant`, `geometry`, and `geography` type default to null. Non-null `initial-default` or `write-default` values are invalid for those four types.

Unknown type

The `unknown` type represents a column whose type has not yet been resolved. It must be optional with null defaults and is not stored in data files, which makes it useful for placeholder columns during schema evolution and when migrating from other table formats. For examples, see the section called “Using the unknown type”.

Version Compatibility

V3 maintains backward compatibility with V2 tables. AWS services support both V2 and V3 tables simultaneously, allowing you to:

- Run queries across both V2 and V3 tables

- Upgrade existing V2 tables to V3 without data rewrites
- Execute time travel queries that span V2 and V3 snapshots
- Use schema evolution and hidden partitioning across table versions

> **Important**
>
> V3 is a one-way upgrade. Once a table is upgraded from V2 to V3, it cannot be downgraded back to V2 through standard operations.

Getting Started with V3

Prerequisites

Before working with V3 tables, ensure you have:

- An AWS account with appropriate IAM permissions
- Access to one or more AWS analytics services (EMR, Glue, Amazon SageMaker Unified Studio Notebooks, or S3 Tables)
- To use the `geometry` and `geography` types in Apache Spark, set `spark.sql.geospatial.enabled=true` in your Spark configuration. On AWS Glue, set it through the job's `--conf` argument. The variant, unknown, nanosecond timestamp, and column default value features need no additional Spark configuration.
- An S3 bucket for storing table data and metadata
- A table bucket to get started with S3 Tables or a general purpose S3 bucket if you are building your own Iceberg infrastructure
- AWS Glue catalog configured

Creating V3 Tables

Creating New V3 Tables

To create a new Iceberg V3 table, set the format-version table property to 3.

Using Spark SQL:

```
CREATE TABLE IF NOT EXISTS myns.orders_v3 (
    order_id bigint,
    customer_id string,
    order_date date,
    total_amount decimal(10,2),
    status string,
    created_at timestamp
)
USING iceberg
TBLPROPERTIES (
    'format-version' = '3'
)
```

Upgrading V2 Tables to V3

You can upgrade existing V2 tables to V3 atomically without rewriting data.

Using Spark SQL:

```
ALTER TABLE myns.existing_table
SET TBLPROPERTIES ('format-version' = '3')
```

> **Important**
>
> V3 is a one-way upgrade. Once a table is upgraded from V2 to V3, it cannot be downgraded back to V2 through standard operations. Before you upgrade, verify that every engine that reads or writes the table supports V3. Amazon Athena can't read V3 tables. For engine support, see Troubleshooting.

What happens during upgrade:

- A new metadata snapshot is created atomically
- Existing Parquet data files are reused
- Row-lineage fields are added to the table metadata
- The next compaction will remove old V2 delete files
- New modifications will use V3's Deletion Vector files

- The upgrade does not perform a historical backfill of row-lineage change tracking records

Enabling Deletion Vectors

To take advantage of Deletion Vectors for updates, deletes, and merges, configure your write mode.

Using Spark SQL:

```
ALTER TABLE myns.orders_v3
SET TBLPROPERTIES ('format-version' = '3',
                   'write.delete.mode' = 'merge-on-read',
                   'write.update.mode' = 'merge-on-read',
                   'write.merge.mode' = 'merge-on-read'
                  )
```

These settings ensure that update, delete, and merge operations create Deletion Vector files instead of rewriting entire data files.

Leveraging Row-lineage for Change Tracking

V3 automatically adds row-lineage metadata fields to track changes.

Using Spark SQL:

```
# Query with parameter value provided
last_processed_sequence = 47

SELECT
    id,
    data,
    _row_id,
    _last_updated_sequence_number
FROM myns.orders_v3
WHERE _last_updated_sequence_number > :last_processed_sequence
```

The _row_id field uniquely identifies each row, while _last_updated_sequence_number tracks when the row was last modified. Use these fields to:

- Identify changed rows for incremental processing
- Track data lineage for compliance
- Optimize CDC pipelines
- Reduce compute costs by processing only changes

Using the variant data type

> **Important**
>
> The variant data type is available only in specific AWS Regions. For the full list of supported Regions, see the section called “Availability”.

With the variant data type, you can write semi-structured data like JSON directly in your Iceberg tables without defining a fixed schema in advance. You can write data faster while still getting efficient analytical query performance. Iceberg V3 compatible engines shred your semi-structured data into hidden columns as you write it. These hidden columns generate Parquet column statistics that query engines use for optimizations like file pruning.

Creating a table with a variant column using Spark SQL:

```
CREATE TABLE IF NOT EXISTS myns.events (
    event_id bigint,
    event_timestamp timestamp,
    source string,
    event_data VARIANT
)
USING iceberg
TBLPROPERTIES (
    'format-version' = '3'
)
```

Inserting semi-structured data into the variant column:

```
INSERT INTO myns.events VALUES (
```

```
    1,
    current_timestamp(),
    'web-app',
    PARSE_JSON('{"user_id": "u-1234", "action": "page_view", "page": "/products",
 "duration_ms": 350}')
);

INSERT INTO myns.events VALUES (
    2,
    current_timestamp(),
    'mobile-app',
    PARSE_JSON('{"user_id": "u-5678", "action": "purchase", "items": [{"sku": "A100",
 "qty": 2}], "total": 49.99}')
);
```

With S3 Tables, table maintenance for variant columns, including compaction, runs automatically. Compaction consolidates data from semi-structured sources from small files into larger files that Iceberg engines can read more efficiently, improving query performance over time.

Using the geospatial data types

Creating a table with geospatial columns using Spark SQL:

```
CREATE TABLE IF NOT EXISTS myns.vehicle_telemetry (
    vehicle_id     string,
    event_time     timestamp,
    position       geometry(4326),
    service_area   geography(4326),
    firmware       string
)
USING iceberg
TBLPROPERTIES ('format-version' = '3')
```

Inserting geospatial values:

```
INSERT INTO myns.vehicle_telemetry VALUES (
    'v-1024',
    TIMESTAMP '2026-09-18 14:22:31.123456',
```

```
    ST_SetSrid(ST_GeomFromWKT('POINT (1 2)'), 4326),
    ST_SetSrid(ST_GeogFromWKT('POLYGON ((0 0, 10 0, 10 10, 0 10, 0 0))'), 4326),
    'fw-3.2.1'
)
```

Filtering on location:

```
SELECT vehicle_id, event_time
FROM myns.vehicle_telemetry
WHERE ST_Intersects(
        position,
        ST_SetSrid(ST_GeomFromWKT('POLYGON ((0 0, 10 0, 10 10, 0 10, 0 0))'), 4326)
      )
```

Note

Existing columns that hold encoded coordinates, such as paired `double` values or WKT strings, are not converted when you upgrade a table to V3. To adopt the geospatial types for existing data, add a column of the new type and populate it.

Note

Geodesic spatial predicates over `geography` are engine-dependent. Spatial predicates in Apache Spark on Amazon EMR and AWS Glue operate on `geometry`.

Using nanosecond-precision timestamps

Creating a table with a nanosecond timestamp column using Spark SQL:

```
CREATE TABLE IF NOT EXISTS myns.trades (
    trade_id     bigint,
    executed_at  timestamptz(9),
    symbol       string,
    price        decimal(18,8)
```

```
)
USING iceberg
TBLPROPERTIES ('format-version' = '3')
```

`timestamp(9)` stores a timestamp without a time zone. `timestamptz(9)` stores one with a time zone.

> **Note**
>
> Reading or writing a nanosecond column with an engine that supports only microsecond precision truncates the value. Confirm that your engine supports nanosecond precision before you rely on it.

Using column default values

Adding a column with a default value using Spark SQL:

```
ALTER TABLE myns.orders
ADD COLUMN currency string DEFAULT 'USD'
```

Rows written before the `currency` column was added return USD rather than null, and no data files are rewritten. The value is stored once in the table's schema metadata as the field's `initial-default` and substituted at read time.

> **Note**
>
> Columns of unknown, `variant`, `geometry`, and `geography` type must default to null, so a non-null default value is invalid for them. For examples of those types, see the section called “Using the unknown type”, the section called “Using the variant data type”, and the section called “Using the geospatial data types”.

Using the unknown type

Creating a table with an unknown column using Spark SQL:

```
CREATE TABLE IF NOT EXISTS myns.staging_events (
    event_id   bigint,
    payload    string,
    reserved   unknown
)
USING iceberg
TBLPROPERTIES ('format-version' = '3')
```

An unknown column must be optional, always reads as null, and is not written to data files. You can later evolve it to a concrete type.

Best Practices for V3

When to Use V3

Consider upgrading to or starting with V3 when:

- You perform frequent batch updates or deletes
- You need to meet GDPR or compliance delete requirements
- Your workloads involve high-frequency upserts
- You require efficient CDC workflows
- You want to reduce storage costs from small files
- You need better change tracking capabilities

Optimizing Write Performance

- Enable Deletion Vectors for update-heavy workloads:

```
SET TBLPROPERTIES (
'write.delete.mode' = 'merge-on-read',
'write.update.mode' = 'merge-on-read',
'write.merge.mode' = 'merge-on-read'
)
```

- Configure appropriate file sizes:

```
SET TBLPROPERTIES (
'write.target-file-size-bytes' = '536870912'  — 512 MB
)
```

Optimizing Read Performance

- Leverage row-lineage for incremental processing
- Use time travel to access historical data without copying
- Enable statistics collection for better query planning

Migration Strategy

When migrating from V2 to V3:

- Test in non-production first - Validate upgrade process and performance
- Upgrade during low-activity periods - Minimize impact on concurrent operations
- Monitor initial performance - Track metrics after upgrade
- Run compaction - Consolidate delete files after upgrade
- Update documentation - Reflect V3 features in team documentation

Compatibility Considerations

- Engine versions - Ensure all engines accessing the table support V3
- Third-party tools - Verify V3 compatibility before upgrading
- Backup strategy - Test snapshot-based recovery procedures
- Monitoring - Update monitoring dashboards for V3-specific metrics

The V3 data types are supported only for tables that use the Parquet data file format. They are not supported for tables that use the ORC or Avro data file formats.

For V3 tables, the sort and Z-order compaction strategies don't support the variant, geometry, geography, and nanosecond-precision timestamp data types.

Considerations for compaction

Compaction writes shredded variant Parquet files by default. Older readers that do not support shredding might fail to read compacted files. You can disable shredding by setting the table property `write.variant.shredding.enabled=false`.

Troubleshooting

Common Issues

Error: "format-version 3 is not supported"

- Check your query engine catalog for compatibility with Iceberg V3.
- Ensure that you are using the latest AWS service versions.
- Verify your engine version supports V3

  V3 support for Amazon AWS services is as follows:

Service	V3 Support	V3 variant support
EMR Spark	Release 7.12+	Release 8.0+
AWS Glue ETL	Version 5.1+	Version 6.0+
Amazon SageMaker Unified Studio Notebooks	Yes	No
AWS Glue: Iceberg REST API, Table Maintenance	Yes	No
Amazon S3 Tables: Iceberg REST API, Table Maintenance	Yes	Yes*
Amazon Athena (Trino)	No	No
Amazon Redshift	Patch 204+	No

*Partial Region availability

S3 Tables support all V3 data types.

Column default values are supported in S3 Tables. They are also supported in AWS Glue version 5.1 and later, and in Amazon Redshift.

Performance degradation after upgrade

- Verify there are no compaction failures. See Logging and monitoring for S3 Tables for more details.
- Check if Deletion Vectors are enabled. Ensure the following properties are set:

```
SET TBLPROPERTIES (
'write.delete.mode' = 'merge-on-read',
'write.update.mode' = 'merge-on-read',
'write.merge.mode' = 'merge-on-read'
)
```

- You can verify table properties with the following code:

```
DESCRIBE FORMATTED myns.orders_v3
```

- Review partition strategy. Over partitioning can lead to small files. Run the below query to get the average file size for your table:

```
SELECT avg(file_size_in_bytes) as avg_file_size_bytes
FROM myns.orders_v3.files
```

Incompatibility with third-party tools

- Verify tool supports V3 specification
- Consider maintaining V2 tables for unsupported tools
- Contact tool vendor for V3 support timeline

Getting Help

- AWS Support: Contact AWS Support for service-specific issues
- Apache Iceberg Community: Iceberg Slack
- AWS Documentation: AWS Analytics Documentation

Pricing

- Amazon EMR: Compute and storage pricing
- Amazon SageMaker pricing
- AWS Glue: Job run and Data Catalog pricing
- S3 Tables: Storage and request pricing

Availability

Apache Iceberg V3 support for deletion vectors and row lineage is available across all AWS Regions where Amazon EMR, AWS Glue Data Catalog, AWS Glue ETL, and S3 Tables operate.

Column default values and the geometry, geography, unknown, and nanosecond-precision timestamp data types are available in all AWS Regions where S3 Tables are available.

The variant data type in S3 Tables is available in the following AWS Regions: US East (N. Virginia), US East (Ohio), US West (Oregon), Asia Pacific (Mumbai), Asia Pacific (Seoul), Asia Pacific (Singapore), Asia Pacific (Sydney), Asia Pacific (Tokyo), Canada (Central), Europe (Frankfurt), Europe (Ireland), Europe (London), Europe (Paris), Europe (Stockholm), and South America (São Paulo).

Additional Resources

- Apache Iceberg V3 Documentation
- Migration Best Practices
- Getting Started Guide

Replicating S3 tables

Amazon S3 Tables support automatic replication of Apache Iceberg tables stored in Amazon S3 table buckets. Replication destinations can be within the same AWS Region, across multiple AWS

Regions, the same account, or to other AWS accounts. By configuring replication for your tables, you can maintain read-only replicas of your data across multiple locations. You can use replicas to enhance data availability, meet compliance requirements and increase access performance for distributed applications.

S3 Tables replication maintains data consistency by committing all table updates, including snapshots, metadata, and data files, to the destination table in the same order as the source table.

When to use S3 Tables replication

You can use S3 Tables replication for the following purposes:

- **Minimize latency** – If your customers are in two geographic locations, you can minimize latency when accessing tables by maintaining read replicas in AWS Regions that are geographically closer to your users.
- **Regulatory compliance** – You can maintain read replicas in specific geographic locations or AWS accounts, which might help you meet certain regulatory or compliance requirements. You can configure the replication destination table bucket to encrypt tables with different AWS KMS keys than the source.
- **Centralized analytics** – If you have data distributed across multiple AWS Regions, you can replicate Region-specific datasets to a centralized Region for unified reporting, cross-Region analysis, and machine learning model training. This eliminates the need to query across Regions or build custom data aggregation pipelines.
- **Testing and development environments** – You can create read replicas of production tables in separate AWS accounts or table buckets to provide realistic test data for development and QA teams. This isolates testing workloads from production systems while ensuring test environments have current, production-like data without manual exports or data synchronization processes.

Features

S3 Tables replication offers the following features.

Read-only replicas for S3 Tables

S3 Tables replication creates read-only replicas of your Apache Iceberg tables across table buckets. You can query replicas independently using any Iceberg-compatible engine.

Automatically maintained replicas

The S3 Tables replication service automatically maintains replica tables. Replication typically updates replicas within minutes of updates to the source. S3 Tables commits all updates in the same order as the source table to maintain consistency.

Replication to multiple destinations

You can replicate the same table to multiple destination table buckets. Replication destinations can be within the same AWS Region, across multiple AWS Regions, in the same AWS account, or in other AWS accounts.

Independent snapshot retention

Snapshot expiration for replica tables is independent of the source table, allowing you to set different retention periods on replica tables if needed. For example, you can configure your source table to retain snapshots for 30 days while setting a 90-day retention period for replica tables. If you configure a longer retention period on replicas, snapshots that expire at the source remain available and queryable in replicas. This configuration provides extended time-travel capabilities for historical analysis.

Maintain replica tables in lower-cost storage tiers

You can configure destination table buckets to use the S3 Intelligent-Tiering storage class, which automatically optimizes storage costs based on access patterns without performance impact or operational overhead. S3 Intelligent-Tiering is well-suited for replica tables that may be accessed less frequently.

For more information about S3 Tables replication, see the following topics.

Topics

- How S3 Tables replication works
- Setting up S3 Tables replication
- Managing S3 Tables replication

How S3 Tables replication works

S3 Tables replication creates read-only replicas of your Apache Iceberg tables across Regions and AWS accounts. Replica tables are maintained automatically by the S3 Tables service and

contain the complete data, metadata, and snapshot history from your source table, making them queryable using any Iceberg-compatible engine for analytics and time-travel operations.

When you configure replication for a table, S3 Tables:

- Creates a read-only replica table in each destination table bucket with the same name and namespace as the source table.
- Backfills the replica with the latest state of the source table
- Monitors the source table for new updates
- Commits all updates to replicas in the same order as the source to maintain consistency

For more information, see the following sections.

Topics

- What is replicated
- How data is replicated
- Snapshot replication
- Considerations and limitations

What is replicated

The following table components are replicated:

- **Table snapshots** – All snapshots, including compacted snapshots, are replicated in chronological order, maintaining parent-child relationships and sequence numbers from the source table. This ensures that replica tables provide the same time-travel capabilities as source tables.
- **Table data** – All data files referenced by table snapshots are replicated to the destination Region. This includes:
  - **Metadata files** – Table metadata.json files, manifests, manifest lists, partition statistics and table statistics.
  - **Delete files** – All delete files are replicated to maintain data accuracy in replica tables.
  - **Data files** – All data files referenced by manifests are replicated.
- **Table metadata** – Complete metadata replication, including schema information (current and historical), partition specifications, sort orders, and table properties.

- **Schema Information** – All table schemas are replicated, including the current schema and historical schema versions. This ensures that queries against replica tables use the correct column definitions, data types, and field mappings. The replication process maintains schema evolution history, allowing time travel queries to work correctly on replica tables.
- **Partition Specifications** – Current and historical partition specifications are replicated, ensuring replica tables maintain the same partitioning strategy as source tables.
- **Sort Orders** – Table sort orders are replicated to maintain query performance optimizations.

How data is replicated

Replication determines a valid state for replica tables by comparing Apache Iceberg table metadata between the source and replica tables. Replication processes metadata in three categories to update your replica table.

For table metadata

For versioned metadata fields, replication merges values from the source table into the replica table's arrays for the following fields:

- `snapshots` – Merges all snapshots from the source table into the replica table's snapshots array by snapshot-id.
- `snapshot-log` – Merges snapshot logs from the source table into the replica table's snapshot-log array, sorted by timestamp and snapshot-id.
- `sort-orders` – Merges sort order definitions from the source table into the replica table's sort-orders array by order-id.
- `partition-specs` – Merges partition specifications from the source table into the replica table's partition-specs array by spec-id.
- `schemas` – Merges schema definitions from the source table into the replica table's schemas array by schema-id.

For table configuration

For fields that represent table configuration, replication copies values directly from the source table:

- `properties`

- `partition-statistics`
- `statistics`

Current table state is also transferred from the source table:

- `current-snapshot-id`
- `current-schema-id`
- `last-column-id`
- `last-partition-id`
- `last-sequence-number`
- `default-sort-order-id`
- `next-row-id` (Iceberg V3)
- `encryption-keys` (Iceberg V3)

Replica-specific state

The following fields are calculated from merged data and updated for the replica table:

- `location` is updated during replication to point to the correct file location in the replica table bucket, ensuring that all file references are valid in the destination environment.
- `metadata-log` contains all destination metadata filenames and is updated after every successful replication with the current metadata filename.
- All file paths are modified to point to the replica table locations.

Snapshot replication

S3 Tables replication maintains complete snapshot history across Regions by replicating all table snapshots in the same commit order as the source table. The parent-child relationships from the source table are preserved in the replica table.

Snapshot retention

You can configure a custom snapshot retention period for your replicated tables that differs from the retention period of the source. This means that even if snapshots are expired and no longer available in the source table, they can be preserved in replicas.

For example, if your source table has a 30-day snapshot retention period but your replica table is configured with a 90-day retention period, the replica will maintain snapshots from the previous two months that are no longer available in the source table.

Snapshots that you manually expired at the source table are also preserved in the replica table. For example, if you expired snapshots from February at the source table using a Spark procedure, you can still time travel to the snapshots in the replica table.

Considerations and limitations

The following considerations apply to replicated tables:

- S3 Tables replicates both Iceberg V2 and V3 tables. However, replication of upgraded tables (V2 → V3) is not supported.
- Metadata files larger than 500MB are not supported.
- While table updates are typically replicated within minutes, replication can take longer depending on the size of the table update to be replicated, for example, when replication starts backfilling.
- Tables with tags or branches are not supported.
- Replication is not supported for Amazon S3 Metadata tables or other AWS-generated system tables.
- All table snapshots, including compacted snapshots, are replicated from the source table. As a result, compaction is not supported on replica tables.

Setting up S3 Tables replication

You can set up replication to automatically create table replicas from a source table to up to five destination table buckets. Replication can be configured at the bucket level (applying to all tables in the bucket) or at the table level (for specific tables). This topic explains how to configure replication using the Amazon S3 console or the AWS Command Line Interface (AWS CLI).

For more information about setting up replication, see the following topics.

Topics

- Prerequisites for setting up replication
- Understanding replication configurations

- Choosing between bucket-level and table-level replication
- Setting up replication by using the Amazon S3 console
- Setting up replication by using the AWS CLI

Prerequisites for setting up replication

Before you configure replication, ensure you have the following:

Required resources

- **Source table bucket** – The table bucket containing the table(s) you want to replicate
- **Destination table bucket(s)** – One or more table buckets where you want to replicate your tables (up to 5 destination table buckets)
- **Source table(s)** – Existing tables in your source table bucket to replicate
- **IAM role(s)** – An IAM role that grants Amazon S3 permissions to replicate tables on your behalf

Required permissions

The IAM identity that you use to set up replication must have the following permissions:

For bucket-level replication:

- `s3tables:PutTableBucketReplication` on the source table bucket
- `s3tables:GetTableBucketReplication` on the source table bucket
- `iam:PassRole` for the replication IAM role

For table-level replication:

- `s3tables:PutTableReplication` on the source table
- `s3tables:GetTableReplication` on the source table
- `iam:PassRole` for the replication IAM role

For cross-account replication:

- Permissions from the destination account's bucket policy

Additional requirements for cross-account replication

If your source and destination table buckets are in different AWS accounts, you also need:

- A bucket policy on the destination table bucket that grants the source account permissions to replicate tables
- The destination account ID and table bucket Amazon Resource Name (ARN)

Additional requirements for encrypted tables

If you want to encrypt replica tables with AWS KMS:

- A KMS key in the destination Region
- Permissions to use the KMS key in your IAM replication role
- A KMS key policy that allows the replication role to encrypt data

Understanding replication configurations

A replication configuration defines how Amazon S3 replicates tables from your source table bucket. Replication can be configured at two levels:

Bucket-level replication

A bucket-level replication configuration applies to all tables in the source table bucket. When you configure bucket-level replication, Amazon S3 automatically replicates any existing tables and any new tables created in the bucket.

Use bucket-level replication when:

- You want to replicate all tables in a bucket
- You want consistent replication behavior across all tables
- You want to simplify management by having a single configuration

Table-level replication

A table-level replication configuration applies to a specific table. Table-level configurations override bucket-level configurations for that specific table.

Use table-level replication when:

- You want to replicate only specific tables
- You need different replication destinations for different tables
- You want to override a bucket-level configuration for certain tables

Replication configuration elements

Each replication configuration contains:

- **IAM role** – The role that Amazon S3 assumes to perform replication operations
- **Rules** – One or more replication rules (limited to 1 rule at launch). Each rule contains:
  - **Destinations** – List of destination table bucket ARNs (up to 5 destinations)
  - **Status** – Whether the rule is enabled or disabled
- **Version token** – A token used to prevent write conflicts when updating configurations

Choosing between bucket-level and table-level replication

Configuration precedence

When both bucket-level and table-level configurations exist:

- Table-level configuration takes precedence for that specific table.
- Other tables follow the bucket-level configuration.

Setting up replication by using the Amazon S3 console

This procedure shows you how to configure replication using the Amazon S3 console.

To set up bucket-level replication

This procedure shows you how to create a table bucket replication configuration using the Amazon S3 console. A table bucket replication configuration applies to all tables in the source table bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Table buckets**.
3. In the **Table buckets** list, choose the name of the table bucket for which you want to configure replication.

4. Choose the **Management** tab.
5. In the **Table bucket replication configuration** section, choose **Create table bucket replication configuration**.
6. In the **Destination** section, configure your replication destinations:

   a. In the **Table bucket ARN** field, enter the ARN of the destination table bucket. The format is: `arn:aws:s3tables:region:account-id:bucket/table-bucket-name`

   Alternatively, choose **Browse S3** to select a table bucket from your account.

   b. (Optional) To add additional destinations, choose **Add destination**. You can add up to 4 more table buckets for a total of 5 destinations.
7. In the **IAM role** section, configure the replication role:

   a. For **IAM role selection method**, choose one of the following options:

   - **Create new IAM role** – Amazon S3 creates a new role with the necessary permissions for replication.
   - **Choose from existing IAM roles** – Select an existing role that has the required replication permissions.
   - **Enter IAM role ARN** – Manually enter the ARN of an existing IAM role.

   b. If you chose **Choose from existing IAM roles**, select a role from the **IAM role** dropdown list.

   c. (Optional) Choose **View** to review the selected role's permissions in the IAM console.
8. Choose **Create replication configuration**.

   After you create the replication configuration, Amazon S3 begins the initial backfill process. You can monitor the replication status in the **Table replication status** section, which displays information about each destination including replication status, destination table ARN, and last replicated metadata.

To set up table-level replication

This procedure shows you how to create a table-level replication configuration using the Amazon S3 console. A table replication configuration applies to a specific table and overrides any bucket-level replication configuration for that table.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Table buckets**.
3. In the **Table buckets** list, choose the name of the table bucket that contains the table you want to replicate.
4. Choose the **Tables** tab.
5. In the **Tables** list, choose the name of the table that you want to replicate.
6. Choose the **Management** tab.
7. In the **Table replication configuration** section, choose **Create table replication configuration**.
8. In the **Destination** section, configure your replication destinations:

   a. In the **Table bucket ARN** field, enter the ARN of the destination table bucket. The format is: `arn:aws:s3tables:`*`region`*`:`*`account-id`*`:bucket/`*`table-bucket-name`*

   Alternatively, choose **Browse S3** to select a table bucket from your account.

   b. (Optional) To add additional destinations, choose **Add destination**. You can add up to 4 more table buckets for a total of 5 destinations.
9. In the **IAM role** section, configure the replication role:

   a. For **IAM role selection method**, choose one of the following options:

   - **Create new IAM role** – Amazon S3 creates a new role with the necessary permissions for replication.
   - **Choose from existing IAM roles** – Select an existing role that has the required replication permissions.
   - **Enter IAM role ARN** – Manually enter the ARN of an existing IAM role.

   b. If you chose **Choose from existing IAM roles**, select a role from the **IAM role** list.

   c. (Optional) Choose **View** to review the selected role's permissions in the IAM console.
10. Choose **Create replication configuration**.

What happens next?

After you create the replication configuration:

- Amazon S3 begins the initial backfill process, creating replica tables in each destination bucket

- The replication status changes to **Replicating** once backfill begins
- You can monitor replication progress on the **Management** tab
- Initial replication time depends on the size of your source table

Setting up replication by using the AWS CLI

This procedure shows you how to configure replication using the AWS CLI. Replace the account IDs, regions, and bucket names with your actual values. Add all destination buckets to the permissions.

Step 1: Create an IAM role for replication

First, create an IAM role that Amazon S3 can assume to replicate your tables.

1. Create a trust policy document that allows S3 Tables to assume the role. Save this as `trust-policy.json`:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "replication.s3tables.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Create the IAM role:

```
aws iam create-role \
    --role-name S3TablesReplicationRole \
    --assume-role-policy-document file://trust-policy.json \
    --description "Role for S3 Tables replication"
```

3. Create a permissions policy that grants replication permissions. Save this as `replication-permissions.json`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3tables:GetTable",
                "s3tables:GetTableMetadataLocation",
                "s3tables:GetTableMaintenanceConfiguration",
                "s3tables:GetTableData"
            ],
            "Resource": "arn:aws:s3tables:us-east-2:111122223333:bucket/amzn-s3-demo-table-bucket-source/table/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3tables:ListTables"
            ],
            "Resource": "arn:aws:s3tables:us-east-2:111122223333:bucket/amzn-s3-demo-table-bucket-source"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3tables:CreateTable",
                "s3tables:CreateNamespace"
            ],
            "Resource": "arn:aws:s3tables:us-east-2:444455556666:bucket/amzn-s3-demo-table-bucket-destination"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3tables:PutTableData",
                "s3tables:GetTableData",
                "s3tables:UpdateTableMetadataLocation",
                "s3tables:PutTableMaintenanceConfiguration"
            ],
```

```
            "Resource": "arn:aws:s3tables:us-east-2:444455556666:bucket/amzn-s3-
demo-table-bucket-destination/table/*"
        }
    ]
}
```

4. Attach the permissions policy to the role:

```
aws iam put-role-policy \
    --role-name S3TablesReplicationRole \
    --policy-name S3TablesReplicationPermissions \
    --policy-document file://replication-permissions.json
```

> **Note**
>
> The permissions policy uses two resource scopes because the actions operate at different levels. `CreateTable` and `CreateNamespace` are bucket-level operations that create resources in the bucket, so they use the bucket ARN without `/table/*`. `PutTableData`, `GetTableData`, `UpdateTableMetadataLocation`, and `PutTableMaintenanceConfiguration` operate on existing tables, so they use the bucket ARN with `/table/*`.

5. (Optional) If using KMS encryption, add KMS permissions to your policy:

```
{
  "Effect": "Allow",
  "Action": [
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Condition": {
    "StringLike": {
      "kms:ViaService": [ "s3.us-east-1.amazonaws.com" ]
    },
    "ArnLike": {
      "kms:EncryptionContext:aws:s3:arn": "arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-table-bucket-source/table/*"
```

```
    }
  },
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "kms:Encrypt",
    "kms:Decrypt",
    "kms:GenerateDataKey"
  ],
  "Condition": {
    "StringLike": {
      "kms:ViaService": [ "s3.us-west-2.amazonaws.com" ]
    },
    "ArnLike": {
      "kms:EncryptionContext:aws:s3:arn": [ "arn:aws:s3tables:us-
west-2:444455556666:bucket/amzn-s3-demo-table-bucket-destination/table/*" ]
    }
  },
  "Resource": "*"
}
```

(Cross-account only) Step 2: Configure destination bucket policy

If you are replicating to a different AWS account, the destination account must grant permissions to the source account.

1. In the destination account, create a bucket policy for the destination table bucket. Save this as `destination-bucket-policy.json`:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/S3TablesReplicationRole"
            },
            "Action": [
```

```
                "s3tables:PutTableData",
                "s3tables:GetTableData",
                "s3tables:UpdateTableMetadataLocation",
                "s3tables:PutTableMaintenanceConfiguration"
            ],
            "Resource": "arn:aws:s3tables:us-east-2:444455556666:bucket/amzn-s3-
demo-table-bucket-cross-account-destination/table/*"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/S3TablesReplicationRole"
            },
            "Action": [
                "s3tables:CreateTable",
                "s3tables:CreateNamespace"
            ],
            "Resource": "arn:aws:s3tables:us-east-2:444455556666:bucket/amzn-s3-
demo-table-bucket-cross-account-destination"
        }
    ]
}
```

2. Apply the policy using the S3 Tables API:

```
aws s3tables put-table-bucket-policy \
    --table-bucket-arn arn:aws:s3tables:us-west-2:444455556666:bucket/amzn-s3-demo-
table-bucket-cross-account-destination \
    --resource-policy file://destination-bucket-policy.json \
    --profile destination-account
```

3. Modify your source KMS key to allow S3 Tables replication and maintenance:

```
{
  "Version": "2012-10-17",
  "Id": "key-consolepolicy-3",
  "Statement": [
        {
            "Sid": "allow replication to decrypt",
            "Effect": "Allow",
```

```
            "Principal": {
                "AWS": "replication_role_arn"
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/SOURCE-KEY-ID"
        },
        {
            "Sid": "allow maintenance",
            "Effect": "Allow",
            "Principal": {
                "Service": "maintenance.s3tables.amazonaws.com"
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/SOURCE-KEY-ID"
        }
  ]
}
```

4. Similarly, add permissions in your destination KMS key policy

```
{
    "Version": "2012-10-17",
    "Id": "key-policy-3",
    "Statement": [
        {
            "Sid": "allow maintenance",
            "Effect": "Allow",
            "Principal": {
                "Service": "maintenance.s3tables.amazonaws.com"
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-west-2:444455556666:key/DESTINATION-KEY-
ID-1"
        },
```

```
        {
            "Sid": "allow replication to encrypt/decrypt",
            "Effect": "Allow",
            "Principal": {
                "AWS": "replication_role_arn"
            },
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-west-2:444455556666:key/DESTINATION-KEY-
ID-1"
        }
    ]
}
```

Step 3: Create a replication configuration

You can use the AWS CLI to create a replication configuration at the table bucket-level or the table-level. For more information, see the following procedures.

Create a bucket-level replication configuration

Use this approach to replicate all tables in a bucket.

1. Create a replication configuration file. Save this as `bucket-replication-config.json`:

   Example: Single destination in same account

```
{
  "role": "arn:aws:iam::111122223333:role/S3TablesReplicationRole",
  "rules": [
    {
      "destinations": [
        {
          "destinationTableBucketARN": "arn:aws:s3tables:us-
west-2:111122223333:bucket/amzn-s3-demo-table-bucket-dr"
        }
      ]
    }
  ]
```

```
}
```

Example: Multiple destinations across regions

```
{
  "role": "arn:aws:iam::111122223333:role/S3TablesReplicationRole",
  "rules": [
    {
      "destinations": [
        {
          "destinationTableBucketARN": "arn:aws:s3tables:us-
west-2:111122223333:bucket/amzn-s3-demo-table-bucket-dr"
        },
        {
          "destinationTableBucketARN": "arn:aws:s3tables:eu-
west-1:111122223333:bucket/amzn-s3-demo-table-bucket-eu"
        },
        {
          "destinationTableBucketARN": "arn:aws:s3tables:ap-
south-1:111122223333:bucket/amzn-s3-demo-table-bucket-apac"
        }
      ]
    }
  ]
}
```

Example: Cross-account replication

```
{
  "role": "arn:aws:iam::111122223333:role/S3TablesReplicationRole",
  "rules": [
    {
      "destinations": [
        {
          "destinationTableBucketARN": "arn:aws:s3tables:us-
east-1:444455556666:bucket/amzn-s3-demo-table-bucket-partner"
        }
      ]
```

```
    }
  ]
}
```

2. Apply the bucket-level replication configuration:

```
aws s3tables put-table-bucket-replication \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket \
    --configuration file://bucket-replication-config.json
```

Expected output:

```
{
  "versionToken": "3HL4kqtJl40Nr8X8gdRQBpUMLUo",
  "status": "Success"
}
```

Create a table-level replication configuration

Use this approach to replicate specific tables or to override bucket-level replication.

1. Create a replication configuration file. Save this as `table-replication-config.json`:

Example: Single table replication

```
{
  "role": "arn:aws:iam::111122223333:role/S3TablesReplicationRole",
  "rules": [
    {
      "destinations": [
        {
          "destinationTableBucketARN": "arn:aws:s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket-analytics-bucket"
        }
```

```
      ]
    }
  ]
}
```

Example: Table with multiple destinations

```
{
  "role": "arn:aws:iam::111122223333:role/S3TablesReplicationRole",
  "rules": [
    {
      "destinations": [
        {
          "destinationTableBucketARN": "arn:aws:s3tables:us-
west-2:111122223333:bucket/amzn-s3-demo-table-bucket-dr"
        },
        {
          "destinationTableBucketARN": "arn:aws:s3tables:eu-
west-1:111122223333:bucket/amzn-s3-demo-table-bucket-eu"
        }
      ]
    }
  ]
}
```

2. Apply the table-level replication configuration:

```
aws s3tables put-table-replication \
    --table-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-
bucket/table/amzn-s3-demo-table-bucket-sales-data \
    --configuration file://table-replication-config.json
```

Expected output:

```
{
  "versionToken": "xT2LZkFZ0UuTC2h8XqtGLx2Ak6M",
```

```
  "status": "Success"
}
```

Managing S3 Tables replication

After you configure S3 Tables replication, you can monitor replica status to verify what's replicated. You can check replication status in the Amazon S3 console on the source table's **Management** tab, or by using the AWS CLI. For more information, see Setting up S3 Tables replication.This topic explains how to monitor replication and understand the different status values that indicate whether replication is completed, in progress, or has failed.

Monitoring replication status

Replication jobs run continuously for your replicated tables. You can query the status of replication with the GetTableReplicationStatus API or view it in the Amazon S3 console.

To get the status of replication by using the AWS CLI

The following example gets the replication status using the GetTableReplicationStatus API.

```
aws s3tables get-table-replication-status \
    --table-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket/table/sales-data
```

Expected output:

```
{
  "sourceTableARN": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket/table/sales-data",
  "destinations": [
    {
      "replicationStatus": "COMPLETED",
      "destinationBucketARN": "arn:aws:s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket",
      "destinationTableARN": "arn:aws:s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-table-bucket/table/sales-data",
```

```
      "lastSuccessfulReplicatedUpdate": {
        "metadataLocation": "latest_table_metadata.json",
        "timestamp": "2025-11-15T14:30:00Z"
      }
    },
    {
      "replicationStatus": "PENDING",
      "destinationBucketARN": "arn:aws:s3tables:eu-west-1:111122223333:bucket/amzn-s3-
demo-table-bucket-eu-bucket",
      "destinationTableARN": "arn:aws:s3tables:eu-west-1:111122223333:bucket/amzn-s3-
demo-table-bucket-eu-bucket/table/sales-data",
      "lastSuccessfulReplicatedUpdate": {
        "metadataLocation": "latest_table_metadata.json",
        "timestamp": "2025-11-15T14:25:00Z"
      }
    }
  ]
}
```

For more information, see get-table-replication-status in the *AWS CLI Command Reference*.

The following example shows a destination with a FAILED status, which includes a `failureMessage` field:

```
{
  "replicationStatus": "FAILED",
  "destinationBucketARN": "arn:aws:s3tables:eu-west-1:444455556666:bucket/amzn-s3-demo-
table-bucket-partner",
  "destinationTableARN": "arn:aws:s3tables:eu-west-1:444455556666:bucket/amzn-s3-demo-
table-bucket-partner/table/sales-data",
  "failureMessage": "Insufficient permissions given to successfully complete
 replication."
}
```

Note

The `get-table-replication` command returns the replication configuration (role, rules, and destinations). The `get-table-replication-status` command returns the

replication status (COMPLETED, PENDING, or FAILED), last replicated metadata, and error messages.

Understanding the response

The response contains the following elements:

- **sourceTableARN** – The ARN of the source table being replicated.
- **destinations** – An array of destination status objects, one for each configured replication destination. Each destination object contains:
  - **replicationStatus** – The current replication status for this destination (COMPLETED, PENDING, or FAILED).
  - **destinationBucketARN** – The ARN of the destination table bucket.
  - **destinationTableARN** – The ARN of the replica table in the destination bucket.
  - **lastSuccessfulReplicatedUpdate** – Information about the most recent successful replication:
    - **metadataLocation** – The Iceberg metadata file name that was last successfully replicated. Compare this with the source table's current metadata location to determine if replication is up to date.
    - **timestamp** – The time when this metadata file was replicated to the destination.
  - **failureMessage** (only present when status is FAILED) – A detailed error message describing why replication failed.

Replication status values

Replication can have three possible statuses for each destination:

- **COMPLETED** – All source table snapshots have been successfully replicated to the destination. The source table's latest metadata location matches the last replicated metadata location.
- **PENDING** – Replication is in progress or new commits are waiting to be replicated. The source table's latest metadata location differs from the last replicated metadata location.
- **FAILED** – The last replication job for this table failed. No new updates are being replicated.

If a destination shows a FAILED status, check the `failureMessage` field for details. Common causes include insufficient permissions, destination bucket not found, or KMS key issues. After you

fix the root cause, the service automatically retries replication. You can monitor recovery as the status transitions from FAILED to PENDING to COMPLETED.

S3 Tables AWS Regions, endpoints, and service quotas

The following sections include the supported AWS Regions and service quotas for S3 Tables.

Topics

- S3 Tables AWS Regions and endpoints
- S3 Tables quotas
- S3 Tables table migration quotas

S3 Tables AWS Regions and endpoints

For a list of Regions S3 Tables is currently available in, see Amazon S3 endpoints. To connect programmatically to an AWS service, you use an endpoint. For more information, see AWS service endpoints.

S3 Tables supports dual-stack endpoints for public access and AWS PrivateLink. Dual-stack endpoints allow you to access S3 tables buckets using the Internet Protocol version 6 (IPv6), in addition to the IPv4 protocol, depending on what your network supports.

S3 Tables dual-stack endpoints use the following naming convention: `s3tables.`*`aws-region`*`.api.aws`

For a complete list of S3 Tables endpoints, see Amazon S3 endpoints.

S3 Tables quotas

Quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account. The following are the quotas for S3 Tables resources. For more Amazon S3 quota information, see Amazon S3 quotas.

Name	Default	Adjustable	Description
Table Buckets	100	To request a quota increase, contact Support.	The number of Amazon S3 table buckets that you

Name	Default	Adjustable	Description
			can create per AWS Region in an account.
Namespaces	10,000	To request a quota increase, contact Support.	The number of Amazon S3 table namespaces that you can create per table bucket.
Tables	10,000	To request a quota increase, contact Support.	The number of Amazon S3 tables that you can create per table bucket.

S3 Tables table migration quotas

The following are the quotas for S3 Tables table migration.

Name	Default	Adjustable	Description
Source objects per migration	1,000,000,000	No	The maximum number of objects in the source table that can be migrated in a single migration.
Column type nesting depth	20	No	The maximum nesting depth of column types, like a map of lists in the source table schema. Columns nested deeper than this limit are reported as unsupported in the

Name	Default	Adjustable	Description
			table compatibility report.
Partitions per source table	100,000	No	The maximum number of partitions in the source table that can be migrated.
Source object size	10 GiB	No	The maximum size of a single source object that can be migrated.
Concurrent migrations per account	10	No	The number of concurrent migrations allowed per account.
Concurrent migrations per destination table	1	No	The number of concurrent migrations allowed per destination table at any time.

Making requests to S3 Tables over IPv6

Amazon S3 supports the ability to access S3 buckets using the Internet Protocol version 6 (IPv6), in addition to the IPv4 protocol, using dual-stack endpoints. Dual-stack endpoints resolve to either an IPv6 endpoint or IPv4 endpoint depending on what your network supports.

The following are some things you should know before trying to access S3 Tables over IPv6:

- The client and the network accessing the table bucket must be enabled to use IPv6.
- Your tables client and your S3 client must both have dual-stack enabled.
- If you use IP address filtering IAM policies they must be updated to handle IPv6 addresses. For more information about managing access permissions with IAM, see the section called “Identity and Access Management (IAM)”.

- When using IPv6, server access log files output IP addresses in an IPv6 format. You need to update existing tools, scripts, and software that you use to parse Amazon S3 log files so that they can parse the IPv6 formatted `Remote IP` addresses. For more information, see the section called "Logging server access".

Getting started making S3 Tables requests over IPv6

When you make a request to a dual-stack endpoint, the table bucket URL resolves to an IPv6 or an IPv4 address depending on what your network supports. If your network prefers IPv4 requests will automatically use IPv4. If your network prefers IPv6, requests will use IPv6. No configuration change is required other than updating your client or application to enable the dual-stack endpoint.

When using the REST API, you directly access an Amazon S3 endpoint by using the endpoint name (URI). You can access S3 Tables and table buckets through a dual-stack endpoint using the following naming convention:

`s3tables.<region>.api.aws`

For a complete list of endpoints for S3 Tables, see Amazon Simple Storage Service endpoints and quotas.

When using the AWS CLI, AWS SDKs, and Iceberg clients, you can use a parameter or flag to change to a dual-stack endpoint. You can also specify the dual-stack endpoint directly as an override of the Amazon S3 endpoint in the config file.

You can enable dual-stack endpoint resolution in SDKs or clients by setting the dual-stack flag using the following command:

```
S3TablesClient client = S3TablesClient.builder()
    .region(Region.US_EAST_1)
    .dualstackEnabled(true)
    .build();
```

To use the dual-stack endpoint in the AWS CLI, see Using dual-stack endpoints from the AWS CLI.

For information on using the dual-stack endpoints for AWS PrivateLink, see, Using the dual-stack endpoints to access tables and table buckets.

Security for S3 Tables

Amazon S3 provides a variety of security features and tools. The following is a list of these features and tools supported by S3 Tables. Proper application of these tools can help ensure that your resources are protected and accessible only to the intended users.

Identity-based policies

Identity-based policies are attached to an IAM user, group, or role. You can use identity-based policies to grant an IAM identity access to your table buckets or tables. By default, users and roles don't have permission to create and modify tables and table buckets. They also can't perform tasks by using S3 console, AWS CLI, or Amazon S3 REST APIs. You can create IAM users, groups, and roles in your account and attach access policies to them. You can then grant access to your resources. To create and access table buckets and tables, an IAM administrator must grant the necessary permissions to the AWS Identity and Access Management (IAM) role or users. For more information, see Access management for S3 Tables.

Resource-based policies

Resource-based policies are attached to a resource. You can create resource-based policies for table buckets and tables. You can use a table bucket policy to control table bucket and namespace-level API access permissions. You can also use a table bucket policy to control table-level API permissions on multiple tables in a bucket. Depending on the policy definition, the permissions attached to the bucket can apply to all or specific tables in the bucket. You can also use a table policy to grant table-level API access permissions to individual tables in the bucket.

When S3 Tables receives a request to perform a table bucket operation or a table operation, it first verifies that the requester has the necessary permissions. It evaluates all the relevant access policies, user policies, and resource-based policies in deciding whether to authorize the request (IAM user policy, IAM role policy, table bucket policy, and table policy). With table bucket policies and table policies, you can personalize access to your resources to ensure that only the identities you have approved can access your resources and perform actions on them. For more information, see Access management for S3 Tables.

AWS Organizations service control policies (SCPs) for S3 Tables.

You can use Amazon S3 Tables in Service Control Policies (SCPs) to manage permissions to users in your organization. Similar to IAM and resource policies, all table and bucket level actions are referenced as part of `s3tables` namespace in the policies. For more information, see Service control policies (SCPs) in the *AWS Organizations User Guide.*

Topics

- Protecting S3 table data with encryption
- Access management for S3 Tables
- VPC connectivity for S3 Tables
- Security considerations and limitations for S3 Tables

Protecting S3 table data with encryption

Using server-side encryption with AWS KMS keys (SSE-KMS) in table buckets

Topics

- How SSE-KMS works for tables and table buckets
- Enforcing and scoping SSE-KMS use for tables and table buckets
- Monitoring and Auditing SSE-KMS encryption for tables and table buckets
- Permission requirements for S3 Tables SSE-KMS encryption
- Specifying server-side encryption with AWS KMS keys (SSE-KMS) in table buckets

Table buckets have a default encryption configuration that automatically encrypts tables by using server-side encryption with Amazon S3 managed keys (SSE-S3). This encryption applies to all tables in your S3 table buckets, and comes at no cost to you.

If you need more control over your encryption keys, such as managing key rotation and access policy grants, you can configure your table buckets to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS). The security controls in AWS KMS can help you meet encryption-related compliance requirements. For more information about SSE-KMS, see Using server-side encryption with AWS KMS keys (SSE-KMS).

How SSE-KMS works for tables and table buckets

SSE-KMS with table buckets differs from SSE-KMS in general purpose buckets in the following ways:

- You can specify encryption settings for table buckets and individual tables.
- You can only use customer managed keys with SSE-KMS. AWS managed keys aren't supported.

- You must grant permissions for certain roles and AWS service principals to access your AWS KMS key. For more information, see Permission requirements for S3 Tables SSE-KMS encryption. This includes granting access to:
  - The S3 maintenance principal – for performing table maintenance on encrypted tables
  - Your S3 Tables integration role – for working with encrypted tables in AWS analytics services
  - Your client access role – for direct access to encrypted tables from Apache Iceberg clients
  - The S3 Metadata principal – for updating encrypted S3 metadata tables
- Encrypted tables use table-level keys that minimize the number of requests made to AWS KMS to make working with SSE-KMS encrypted tables more cost effective.

SSE-KMS encryption for table buckets

When you create a table bucket, you can choose SSE-KMS as the default encryption type and select a specific KMS key that will be used for encryption. Any tables created within that bucket will automatically inherit these encryption settings from their table bucket. You can use the AWS CLI, S3 API, or AWS SDKs to modify or remove the default encryption settings on a table bucket at any time. When you modify the encryption settings on a table bucket those settings apply only to new tables created in that bucket. Encryption settings for pre-existing tables are not changed. For more information, see Specifying encryption for table buckets.

SSE-KMS encryption for tables

You also have an option to encrypt an individual table with a different KMS key regardless of the bucket's default encryption configuration. To set encryption for an individual table, you must specify the desired encryption key at the time of table creation. If you want to change the encryption for an existing table, then you'll need to create a table with desired key and copy data from old table to the new one. For more information, see Specifying encryption for tables.

When using AWS KMS encryption, S3 Tables automatically creates unique table-level data keys that encrypt new objects associated with each table. These keys are used for a limited time period, minimizing the need for additional AWS KMS requests during encryption operations and reducing the cost of encryption. This is similar to S3 Bucket Keys for SSE-KMS.

Enforcing and scoping SSE-KMS use for tables and table buckets

You can use S3 Tables resource-based policies, KMS key policies, IAM identity-based policies, or any combination of these, to enforce the use of SSE-KMS for S3 tables and table buckets. For more

information on identity and resource polices for tables, see Access management for S3 Tables. For information on writing key policies, see Key policies in the *AWS Key Management Service Developer Guide*. The following examples show how you can use policies to enforce SSE-KMS.

Enforcing the use of SSE-KMS for all tables with a table bucket policy

This is an example of table bucket policy that prevents users from creating tables in a specific table bucket unless they encrypt tables with a specific AWS KMS key. To use this policy, replace the *`user input placeholders`* with your own information:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceKMSEncryptionAlgorithm",
      "Effect": "Deny",
      "Principal": "*",
      "Action": [
        "s3tables:CreateTable"
      ],
      "Resource": [
        "arn:aws:s3tables:us-west-2:111122223333:bucket/example-table-bucket/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "s3tables:sseAlgorithm": "aws:kms"
        }
      }
    },
    {
      "Sid": "EnforceKMSEncryptionKey",
      "Effect": "Deny",
      "Principal": "*",
      "Action": [
        "s3tables:CreateTable"
      ],
      "Resource": [
        "arn:aws:s3tables:us-west-2:111122223333:bucket/example-table-bucket/*"
      ],
      "Condition": {
        "StringNotEquals": {
```

```
          "s3tables:kmsKeyArn": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
      }
    }
  ]
}
```

Requiring users to use SSE-KMS encryption with an IAM policy

This IAM identity policy requires users to use a specific AWS KMS key for encryption when creating or configuring S3 Tables resources. To use this policy, replace the *user input placeholders* with your own information:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RequireSSEKMSOnTables",
      "Action": [
          "s3tables:CreateTableBucket",
          "s3tables:PutTableBucketEncryption",
          "s3tables:CreateTable"
      ],
      "Effect": "Deny",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
            "s3tables:sseAlgorithm": "aws:kms"
        }
      }
    },
    {
      "Sid": "RequireKMSKeyOnTables",
      "Action": [
          "s3tables:CreateTableBucket",
          "s3tables:PutTableBucketEncryption",
          "s3tables:CreateTable"
      ],
      "Effect": "Deny",
      "Resource": "*",
      "Condition": {
```

```
        "StringNotEquals": {
            "s3tables:kmsKeyArn": "<key_arn>"
        }
      }
    }
  ]
}
```

Restricting the use of a key to a specific table bucket with a KMS key policy

This example KMS key policy allows the key to be used by a specific user only for encryption operations in a specific table bucket. This type of policy is useful for limiting access to a key in cross-account scenarios. To use this policy, replace the *user input placeholders* with your own information:

JSON

```
{
  "Version":"2012-10-17",
  "Id": "Id",
  "Statement": [
    {
      "Sid": "AllowPermissionsToKMS",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:root"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "<table-bucket-arn>/*"
        }
      }
    }
  ]
}
```

Monitoring and Auditing SSE-KMS encryption for tables and table buckets

To audit the usage of your AWS KMS keys for your SSE-KMS encrypted data, you can use AWS CloudTrail logs. You can get insight into your cryptographic operations, such as `GenerateDataKey` and `Decrypt`. CloudTrail supports numerous attribute values for filtering your search, including event name, user name, and event source.

You can track encryption configuration requests for Amazon S3 tables and table buckets by using CloudTrail events. The following API event names are used in CloudTrail logs:

- `s3tables:PutTableBucketEncryption`
- `s3tables:GetTableBucketEncryption`
- `s3tables:DeleteTableBucketEncryption`
- `s3tables:GetTableEncryption`
- `s3tables:CreateTable`
- `s3tables:CreateTableBucket`

> **Note**
>
> EventBridge isn't supported for table buckets.

Permission requirements for S3 Tables SSE-KMS encryption

When you use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) for tables in S3 table buckets you need to grant permissions for different identities in your account. At minimum your access identity and the S3 Tables maintenance principal need access to your key, the other permissions required depend on your use case.

Required Permissions

To access a table encrypted with a KMS key, you need these permissions on that key:

- `kms:GenerateDataKey`
- `kms:Decrypt`

> **Important**
>
> To use SSE-KMS on tables the Amazon S3 Tables maintenance service principal (`maintenance.s3tables.amazonaws.com`) needs `kms:GenerateDataKey` and `kms:Decrypt` permissions on the key.

Additional permissions

These additional permissions are required depending on your use case:

- **Permissions for AWS analytics services and direct access** – If you work with SSE-KMS encrypted tables through AWS analytics services or third-party engines accessing S3 tables directly, the IAM role you use needs permission to use your KMS key.
- **Permissions with Lake Formation enabled** – If you have opted in to AWS Lake Formation for access control, the Lake Formation service role needs permission to use your KMS key.
- **Permissions for S3 Metadata tables** – If you use SSE-KMS encryption for S3 Metadata tables, you need to provide the S3 Metadata service principal (`metadata.s3.amazonaws.com`) access to your KMS key. This allows S3 Metadata to update encrypted tables so they will reflect your latest data changes.

> **Note**
>
> For cross-account KMS keys, your IAM role needs both key access permission and explicit authorization in the key policy. For more information about cross-account permissions for KMS keys, see Allowing external AWS accounts to use a KMS key in the *AWS Key Management Service Service Developer Guide*.

Topics

- Granting the S3 Tables maintenance service principal permissions to your KMS key
- Granting IAM principals permissions to work with encrypted tables in integrated AWS analytics services
- Granting IAM principals permissions to work with encrypted tables when Lake Formation is enabled
- Granting the S3 Metadata service principal permissions to use your KMS key

Granting the S3 Tables maintenance service principal permissions to your KMS key

This permission is required to create SSE-KMS encrypted tables and to allow automatic table maintenance like compaction, snapshot management, and unreferenced file removal on the encrypted tables.

> **Note**
>
> Whenever you make a request to create an SSE-KMS encrypted table, S3 Tables checks to make sure the `maintenance.s3tables.amazonaws.com` principal has access to your KMS key. To perform this check, a zero-byte object is temporarily created in your table bucket, this object will be automatically removed by the unreferenced file removal maintenance operations. If the KMS key you specified for encryption doesn't have maintenance access the createTable operation will fail.

To grant maintenance access on SSE-KMS encrypted tables, you can use the following example key policy. In this policy, the `maintenance.s3tables.amazonaws.com` service principal is granted permission to use a specific KMS key for encrypting and decrypting tables in a specific table bucket. To use the policy, replace the *`user input placeholders`* with your own information:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EnableKeyUsage",
            "Effect": "Allow",
            "Principal": {
                "Service": "maintenance.s3tables.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-id",
            "Condition": {
                "StringLike": {
```

```
                    "kms:EncryptionContext:aws:s3:arn":"<table-or-table-bucket-
arn>/*"
                }
            }
        }
    ]
}
```

Granting IAM principals permissions to work with encrypted tables in integrated AWS analytics services

To work with S3 tables in AWS analytics services, you integrate your table buckets with AWS Glue Data Catalog. This integration allows AWS analytics services to automatically discover and access table data. For more information on the integration, see Integrating Amazon S3 Tables with AWS analytics services.

When you work with SSE-KMS encrypted tables through AWS analytics services or third-party and open-source engines accessing S3 tables directly, the IAM role you use needs permission to use your AWS KMS key for encryption operations.

You can grant KMS key access through an IAM policy attached to your role or through a KMS key policy.

IAM policy

Attach this inline policy to the IAM role you use for querying to allow KMS key access. Replace the KMS key ARN with your own.

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowKMSKeyUsage",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
        }
```

```
    ]
}
```

KMS key policy

Alternatively, attach this statement to your KMS key policy to allow the specified IAM role to use the key. Replace the role ARN with the IAM role you use for querying.

```
{
    "Sid": "Allow use of the key",
    "Effect": "Allow",
    "Principal": {
        "AWS": [
            "arn:aws:iam::<catalog-account-id>:role/<role-name>"
        ]
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey",
    ],
    "Resource": "*"
}
```

Granting IAM principals permissions to work with encrypted tables when Lake Formation is enabled

If you have opted in to AWS Lake Formation for access control on your S3 Tables integration, the Lake Formation service role needs permission to use your AWS KMS key for encryption operations. Lake Formation uses this role to vend credentials on behalf of principals accessing your tables.

The following KMS key policy example grants the Lake Formation service role permission to use a specific KMS key in your account for encryption operations. Replace the placeholder values with your own.

```
{
  "Sid": "AllowTableRoleAccess",
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::111122223333:role/service-role/S3TablesRoleForLakeFormation"
  },
  "Action": [
```

```
      "kms:GenerateDataKey",
      "kms:Decrypt"
  ],
  "Resource": "<kms-key-arn>"
}
```

Granting the S3 Metadata service principal permissions to use your KMS key

To allow Amazon S3 to update SSE-KMS encrypted metadata tables, and perform maintenance on those metadata tables, you can use the following example key policy. In this policy, you allow the `metadata.s3.amazonaws.com` and `maintenance.s3tables.amazonaws.com` service principals to encrypt and decrypt tables in a specific table bucket using a specific key. To use the policy, replace the *`user input placeholders`* with your own information:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EnableKeyUsage",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "maintenance.s3tables.amazonaws.com",
                    "metadata.s3.amazonaws.com"
                ]
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn":"<table-or-table-bucket-
arn>/*"
                }
            }
        }
    ]
```

```
}
```

Specifying server-side encryption with AWS KMS keys (SSE-KMS) in table buckets

All Amazon S3 table buckets have encryption configured by default, and all new tables created in an table bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every table bucket. If you want to specify a different encryption type, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS).

You can specify SSE-KMS encryption in your `CreateTableBucket` or `CreateTable` requests, or you can set the default encryption configuration in the table bucket in a `PutTableBucketEncryption` request.

> **Important**
>
> To allow automatic maintenance on SSE-KMS encrypted tables and table buckets you must grant the maintenance.s3tables.amazonaws.com service principal permission to use your KMS key. For more information, see Permission requirements for S3 Tables SSE-KMS encryption.

Specifying encryption for table buckets

You can specify SSE-KMS as the default encryption type when you create a new table bucket, for examples, see Creating a table bucket. After creating a table bucket, you can specify the use of SSE-KMS as the default encryption setting using REST API operations, AWS SDKs, and the AWS Command Line Interface (AWS CLI).

> **Note**
>
> When you specify SSE-KMS as the default encryption type, the key you use for encryption must allow access to the S3 Tables maintenance service principal. If the maintenance service principal does not have access, you will be unable to create tables in that table bucket. For more information, see Granting the S3 Tables maintenance service principal permissions to your KMS key.

Using the AWS CLI

To use the following example AWS CLI command, replace the *user input placeholders* with your own information.

```
aws s3tables put-table-bucket-encryption \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket; \
    --encryption-configuration '{
        "sseAlgorithm": "aws:kms",
        "kmsKeyArn": "arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
    }' \
    --region us-east-1
```

You can remove the default encryption setting for a table bucket using the DeleteTableBucketEncryption API operation. When you remove encryption settings new tables created in the table bucket will use the default SSE-S3 encryption.

Specifying encryption for tables

You can apply SSE-KMS encryption to a new table when you create it using query engines, REST API operations, AWS SDKs, and the AWS Command Line Interface (AWS CLI). The encryption settings you specify when creating a table take precedence over the default encryption setting of the table bucket.

> **Note**
>
> When you use SSE-KMS encryption for a table the key you use for encryption must allow the S3 Tables maintenance service principal to access it. If the maintenance service principal does not have access, you will be unable to create the table. For more information, see Granting the S3 Tables maintenance service principal permissions to your KMS key.

Required permissions

The following permissions are required to create encrypted tables

- `s3tables:CreateTable`

- `s3tables:PutTableEncryption`

Using the AWS CLI

The following AWS CLI example creates a new table with a basic schema, and encrypts it with a customer managed AWS KMS key. To use the command, replace the *`user input placeholders`* with your own information.

```
aws s3tables create-table \
  --table-bucket-arn "arn:aws:s3tables:Region:ownerAccountId:bucket/amzn-s3-demo-table-bucket" \
  --namespace "mydataset" \
  --name "orders" \
  --format "ICEBERG" \
  --encryption-configuration '{
    "sseAlgorithm": "aws:kms",
    "kmsKeyArn":
 "arn:aws:kms:Region:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }' \
  --metadata '{
    "iceberg": {
      "schema": {
        "fields": [
          {
            "name": "order_id",
            "type": "string",
            "required": true
          },
          {
            "name": "order_date",
            "type": "timestamp",
            "required": true
          },
          {
            "name": "total_amount",
            "type": "decimal(10,2)",
            "required": true
          }
        ]
      }
    }
  }'
```

Data protection refers to protecting data while it's in transit (as it travels to and from Amazon S3) and at rest (while it's stored on disks in Amazon S3 data centers). S3 Tables always protects data in transit using Transport Layer Security (1.2 and above) through HTTPS. For protecting data at rest in S3 table buckets, you have the following options:

Server-side encryption with Amazon S3 managed keys (SSE-S3)

All Amazon S3 table buckets have encryption configured by default. The default option for server-side encryption is with Amazon S3 managed keys (SSE-S3). This encryption comes at no cost to you and applies to all tables in your S3 table buckets unless you specify another form of encryption. Each object is encrypted with a unique key. As an additional safeguard, SSE-S3 encrypts the key itself with a root key that it regularly rotates. SSE-S3 uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data.

Server-side encryption with AWS KMS keys (SSE-KMS)

You can choose to configure table buckets or tables to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS). The security controls in AWS KMS can help you meet encryption-related compliance requirements. SSE-KMS gives you more control over your encryption keys by allowing you to do the following:

- Create, view, edit, monitor, enable or disable, rotate, and schedule deletion of KMS keys.
- Define the policies that control how and by whom KMS keys can be used.
- Track key usage in AWS CloudTrail to verify your KMS keys are being used correctly.

S3 Tables supports using customer managed keys in SSE-KMS to encrypt tables. AWS managed keys are not supported. For more information on using SSE-KMS for S3 tables and table buckets, see Using server-side encryption with AWS KMS keys (SSE-KMS) in table buckets.

Access management for S3 Tables

In S3 Tables resources include table buckets and the tables that they contain. The root user of the AWS account that created the resource (the resource owner) and AWS Identity and Access Management (IAM) users within that account that have the necessary permissions can access a resource that they created. The resource owner specifies who else can access the resource and the actions that they are allowed to perform on the resource. Amazon S3 has various access management tools that you can use to grant others access to your S3 resources. If you've integrated your tables with AWS Lake Formation, then you can also manage fine-grained access to your tables and namespaces. The following topics provide you with an overview of resources, IAM

actions, and condition keys for S3 Tables. They also provide examples for both resource-based and identity-based policies for S3 Tables.

Topics

- Resources
- Actions for S3 Tables
- Condition keys for S3 Tables
- IAM identity-based policies for S3 Tables
- Resource-based policies for S3 Tables
- AWS managed policies for S3 Tables
- Granting access with SQL semantics
- Managing access to a table or database with Lake Formation

Resources

S3 Tables resources include table buckets and the tables that they contain.

- Table buckets – Table buckets are specifically designed for tables and provider higher transactions per seconds (TPS) and better query throughput compared to self-managed tables in general purpose S3 buckets. Table buckets deliver the same durability, availability, scalability, and performance characteristics as Amazon S3 general purpose buckets.
- Tables – Tables in your table buckets are stored in Apache Iceberg format. You can query these tables using standard SQL in query engines that support Iceberg.

Amazon Resource Names (ARNs) for tables and table buckets contain the `s3tables` namespace, the AWS Region, the AWS account ID, and the bucket name. To access and perform actions on your tables and table buckets, you must use the following ARN formats:

- Table ARN format:

```
arn:aws:s3tables:us-west-2:111122223333:bucket/amzn-s3-demo-bucket/
table/demo-tableID
```

Actions for S3 Tables

In an identity-based policy or resource-based policy, you define which S3 Tables actions are allowed or denied for specific IAM principals. Tables actions correspond to bucket and table-level API operations. All actions are part of a unique IAM namespace: `s3tables`.

When you use an action in a policy, you usually allow or deny access to the API operation with the same name. However, in some cases, a single action controls access to more than one API operation. For example, the `s3tables:GetTableData` actions includes permissions for the `GetObject`, `ListParts`, and `ListMultiparts` API operations.

The following are supported actions for table buckets. You can specify the following actions in the `Action` element of an IAM policy or resource policy.

Action	Description	Access level	Cross-account access
`s3tables:CreateTableBucket`	Grants permissions to create a table bucket	`Write`	No
`s3tables:GetTableBucket`	Grants permission to retrieve a table bucket ARN, table bucket name, and create date.	`Write`	Yes
`s3tables:ListTableBuckets`	Grants permission to list all table buckets in this account.	`Read`	No

Action	Description	Access level	Cross-account access
s3tables:CreateNamespace	Grants permission to create a name space in a table bucket	Write	Yes
s3tables:GetNamespace	Grants permission to retrieve namespace details	Read	Yes
s3tables:ListNamespaces	Grants permission to list all namespaces on the table bucket.	Read	Yes
s3tables:DeleteNamespace	Grants permission to delete a namespace in a table bucket	Write	Yes
s3tables:DeleteTableBucket	Grants permission to delete the bucket	Write	Yes

Action	Description	Access level	Cross-account access
s3tables:PutTableBucketPolicy	Grants permission to add or replace a bucket policy	Permissions Management	No
s3tables:GetTableBucketPolicy	Grants permission to retrieve the bucket policy	Read	No
s3tables:DeleteTableBucketPolicy	Grants permission to delete the bucket policy	Permissions Management	No
s3tables:GetTableBucketMaintenanceConfiguration	Grants permission to retrieve the maintenance configuration for a table bucket	Read	Yes
s3tables:PutTableBucketMaintenanceConfiguration	Grants permission to add or replace the maintenance configuration for a table bucket	Write	Yes

Action	Description	Access level	Cross-account access
s3tables:PutTableBucketEncryption	Grants permission to add or replace the encryption configuration for a table bucket	Write	No
s3tables:GetTableBucketEncryption	Grants permission to retrieve the encryption configuration for a table bucket	Read	No
s3tables:DeleteTableBucketEncryption	Grants permission to delete the encryption configuration for a table bucket	Write	No

The following actions are supported for tables:

Action	Description	Access level	Cross-account access
s3tables:GetTableMaintenanc	Grants permission to retrieve the maintenance	Read	Yes

Action	Description	Access level	Cross-account access
eConfigur ation	configuration for a table		
s3tables: PutTableM aintenanc eConfigur ation	Grants permissio n to add or replace the maintenance configuration for a table	Write	Yes
s3tables: PutTableP olicy	Grants permissio n to add or replace a table policy	Permissions Management	No
s3tables: GetTableP olicy	Grants permission to retrieve the table policy	Read	No
s3tables: DeleteTab lePolicy	Grants permission to delete the table policy	Permissions management	No
s3tables: CreateTable	Grants permissio n to create a table in a table bucket	Write	Yes

Action	Description	Access level	Cross-account access
s3tables:GetTable	Grants permission to retrieve a table information	Read	Yes
s3tables:GetTableMetadataLocation	Grants permission to retrieve the table root pointer (metadata file)	Read	Yes
s3tables:ListTables	Grants permission to list all tables in a table bucket	Read	Yes
s3tables:RenameTable	Grant permissions to change the name of a table.	Write	Yes
s3tables:UpdateTableMetadataLocation	Grants permission to update table root pointer (metadata file)	Write	Yes

Action	Description	Access level	Cross-account access
s3tables:GetTableData	Grants permission to read the table metadata and data objects stored in the table bucket	Read	Yes
s3tables:PutTableData	Grants permission to write the table metadata and data objects stored in the table bucket	Write	Yes
s3tables:GetTableEncryption	Grants permission to retrieve the encryption settings for a table	Write	No
s3tables:PutTableEncryption	Grants permission to add encryption to a table	Write	No

Action	Description	Access level	Cross-account access
s3tables:DeleteTable	Grants permission to delete a table from a table bucket	Write	Yes

To perform table-level read and write actions, S3 Tables supports Amazon S3 API operations such as GetObject and PutObject. The following table provides a list of object-level actions. When granting read and write permissions to your tables, you use the following actions.

Action	S3 object APIs		
s3tables:GetTableData	GetObject, ListParts, HeadObject		
s3tables:PutTableData	PutObject, CreateMultipartUpload, CompleteMultipartUpload, UploadPart, AbortMultipartUpload		

For example, if a user has `GetTableData` permissions, then they can read all the files associated with the table, such as its metadata file, manifest, manifest list files, and parquet data files.

Condition keys for S3 Tables

S3 Tables supports AWS global condition context keys.

Additionally, S3 Tables defines the following condition keys that you can use in an access policy.

Condition key	Description	Type	
`s3tables:tableName`	Filters access by the name of the tables in the table bucket. You can use the `s3tables:tableName` condition key to write IAM, or table bucket policies that restrict user or application access to only the tables that meet this name condition. It's important to note that if you use the	`String`	

Condition key	Description	Type	
	`s3tables:tableName` condition key to control access then changes in tables' name could impact these policies. **Example value:** `"s3tables:tableName":"department*"`		

Condition key	Description	Type	
`s3tables:namespace`	Filters access by the namespaces created in the table bucket. You can use the `s3tables:namespace` condition key to write IAM, table, or table bucket policies that restrict user or application access to tables that are part of a specific namespace. *Example value:* `"s3tables:namespace":"hr"` It's important to note that if you use the `s3tables:namespace` condition	`String`	

Condition key	Description	Type	
	key to control access, then changes in namespaces could impact these policies.		

Condition key	Description	Type	
`s3tables:SSEAlgorithm`	Filters access by the server-side encryption algorithm used to encrypt a table. You can use the `s3tables:SSEAlgorithm` condition key to write IAM, table, or table bucket policies that restrict user or application access to tables that are encrypted with a certain encryption type. *Example value:* `"s3tables:SSEAlgorithm":"aws:kms"`	`String`	

Condition key	Description	Type	
	It's important to note that if you use the `s3tables:SSEAlgorithm` condition key to control access, then changes in encryption could impact these policies.		

Condition key	Description	Type	
`s3tables:KMSKeyArn`	Filters access by the AWS KMS key ARN for the key used to encrypt a table You can use the `s3tables:KMSKeyArn` condition key to write IAM, table, or table bucket policies that restrict user or application access to tables that are encrypted with a specific KMS key. It's important to note that if you use the `s3tables:KMSKeyArn` condition key to control	ARN	

Condition key	Description	Type	
	access, then changing your KMS key could impact these policies.		

IAM identity-based policies for S3 Tables

By default, users and roles don't have permission to create or modify tables and table buckets. They also can't perform tasks by using the s3 console, AWS Command Line Interface (AWS CLI), or Amazon S3 REST APIs. To create and access table buckets and tables, an AWS Identity and Access Management (IAM) administrator must grant the necessary permissions to the IAM role or users. To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

The following topic includes examples of IAM identity-based policies. To use the following example policies, replace the *`user input placeholders`* with your own information.

Topics

- Example 1: Allow access to create and use table buckets
- Example 2: Allow access to create and use tables in a table bucket

Example 1: Allow access to create and use table buckets

.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowBucketActions",
            "Effect": "Allow",
            "Action": [
```

```
                "s3tables:CreateTableBucket",
                "s3tables:PutTableBucketPolicy",
                "s3tables:GetTableBucketPolicy",
                "s3tables:ListTableBuckets",
                "s3tables:GetTableBucket"
            ],
            "Resource": "arn:aws:s3tables:us-east-1:111122223333:bucket/*"
        }
    ]
}
```

Example 2: Allow access to create and use tables in a table bucket

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowBucketActions",
            "Effect": "Allow",
            "Action": [
                "s3tables:GetTableBucket",
                "s3tables:ListTables",
                "s3tables:CreateTable",
                "s3tables:PutTableData",
                "s3tables:GetTableData",
                "s3tables:GetTable",
                "s3tables:GetTableMetadataLocation",
                "s3tables:UpdateTableMetadataLocation",
                "s3tables:GetNamespace",
                "s3tables:CreateNamespace",
                "s3tables:ListNamespaces"
            ],
            "Resource": [
                "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket",
                "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket/table/*"
            ]
        }
    ]
```

```
}
```

Resource-based policies for S3 Tables

S3 Tables provides resource-based policies for managing access to table buckets and tables: table bucket policies and table policies. You can use a table bucket policy to grant API access permissions at the table bucket, namespace, or table-level. The permissions attached to the table bucket can apply to all tables in the bucket or to specific tables in the bucket, depending on the policy definition. You can use a table policy to grant permissions at the table-level.

When S3 Tables receives a request, it first verifies that the requester has the necessary permissions. It evaluates all the relevant access policies, user policies, and resource-based policies in deciding whether to authorize the request (IAM user policy, IAM role policy, table bucket policy, and table policy). For example, if a table bucket policy grants a user permissions to perform all actions on the tables in the bucket (including `DeleteTable`), but an individual table has a table policy that denies the `DeleteTable` action for all users, then the user cannot delete the table.

The following topic includes examples of table and table bucket policies. To use these policies, replace the *`user input placeholders`* with your own information.

Note

- Every policy that grants permissions to modify tables should include permissions for `GetTableMetadataLocation` to access the table root file. For more information, see `GetTableMetadataLocation`.
- Every time that you perform a write or delete activity on your table, include permissions to `UpdateTableMetadataLocation` in your access policy.
- We recommend using a table bucket policy for governing access to bucket-level actions and a table policy for governing access to table-level actions. In cases where you want to define the same set of permissions across multiple tables, then we recommend using a table bucket policy.

Topics

- Example 1: Table bucket policy allows access to PutBucketMaintenanceConfiguration for buckets in an account

- Example 2: Table bucket policy to allows read (SELECT) access to tables stored in the hrnamespace
- Example 3: Table policy to allow user to delete a table

Example 1: Table bucket policy allows access to PutBucketMaintenanceConfiguration for buckets in an account

The following example table bucket policy allows the IAM `data steward` to delete unreferenced objects for all the buckets in an account by allowing access to `PutBucketMaintenanceConfiguration`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/datasteward"
            },
            "Action": [
                "s3tables:PutTableBucketMaintenanceConfiguration"
            ],
            "Resource": "arn:aws:s3tables:us-east-1:111122223333:bucket/*"
        }
    ]
}
```

Example 2: Table bucket policy to allows read (SELECT) access to tables stored in the hrnamespace

The following an example table bucket policy allows Jane, a user from AWS account ID `123456789012` to access tables stored in the `hr` namespace in a table bucket.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Jane"
            },
            "Action": [
                "s3tables:GetTableData",
                "s3tables:GetTableMetadataLocation"
            ],
            "Resource": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-
demo-table-bucket/table/*",
            "Condition": {
                "StringLike": {
                    "s3tables:namespace": "hr"
                }
            }
        }
    ]
}
```

Example 3: Table policy to allow user to delete a table

The following example table policy that allows the IAM role `data steward` to delete a table.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "DeleteTable",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/datasteward"
            },
            "Action": [
                "s3tables:DeleteTable",
                "s3tables:UpdateTableMetadataLocation",
                "s3tables:PutTableData",
                "s3tables:GetTableMetadataLocation"
```

```
            ],
            "Resource": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-
demo-bucket/table/tableUUID"
        }
    ]
}
```

AWS managed policies for S3 Tables

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AmazonS3TablesFullAccess

You can attach the `AmazonS3TablesFullAccess` policy to your IAM identities. This policy grants permissions that allow full access to Amazon S3 Tables. For more information about this policy, see AmazonS3TablesFullAccess.

AWS managed policy: AmazonS3TablesReadOnlyAccess

You can attach the `AmazonS3TablesReadOnlyAccess` policy to your IAM identities. This policy grants permissions that allow read-only access to Amazon S3 Tables. For more information about this policy, see AmazonS3TablesReadOnlyAccess.

AWS managed policy: AmazonS3TablesLakeFormationServiceRole

You can attach the `AmazonS3TablesLakeFormationServiceRole` policy to your IAM identities. This policy grants permissions that allow the AWS Lake Formation service role access to S3 Tables. AWS KMS permissions are used to allow Lake Formation to access encrypted tables. For more information about this policy, see AmazonS3TablesLakeFormationServiceRole.

Amazon S3 Tables updates to AWS managed policies

View details about updates to AWS managed policies for Amazon S3 Tables since S3 Tables began tracking these changes.

Change	Description	Date
Amazon S3 Tables updated `AmazonS3TablesFullAccess` .	S3 Tables updated the AWS-managed policy called `AmazonS3TablesFullAccess` . This update grants permission to pass a role to the S3 Tables Replication service.	December 2, 2025
Amazon S3 Tables added `AmazonS3TablesLakeFormationServiceRole` .	S3 Tables added a new AWS-managed policy called `AmazonS3TablesLakeFormationServiceRole` . This policy grants permissions that allows the Lake Formation service role access to S3 Tables.	May 19, 2025
Amazon S3 Tables added `AmazonS3TablesFullAccess` .	S3 Tables added a new AWS-managed policy called `AmazonS3TablesFullAccess` . This policy grants permissions that allow full access to Amazon S3 Tables.	December 03, 2024

Change	Description	Date
Amazon S3 Tables added `AmazonS3TablesReadOnlyAccess` .	S3 Tables added a new AWS-managed policy called `AmazonS3TablesReadOnlyAccess` . This policy grants permissions to allow read-only access to Amazon S3 Tables.	December 03, 2024
Amazon S3 Tables started tracking changes.	Amazon S3 Tables started tracking changes for its AWS managed policies.	December 03, 2024

Granting access with SQL semantics

You can grant permissions to tables by using SQL semantics in table and table bucket policies. Examples of SQL semantics you can use are `CREATE`, `INSERT`, `DELETE`, `UPDATE`, and `ALTER`. The following table provides a list of API actions associated with SQL semantics that you can use to grant permissions to your users.

S3 Tables partially supports permissions using SQL semantics. For example, the `CreateTable` API only creates an empty table in the table bucket. You need additional permissions such as, `UpdateTableMetadata`, `PutTableData`, and `GetTableMetadataLocation` to be able to set the table schema. These additional permissions also mean that you are also granting the user access to insert rows in the table. If you wish to govern access purely based on SQL semantics, then we recommend using AWS Lake Formation or any third-party solution that is integrated with S3 Tables.

Table-level activity	IAM actions		
`SELECT`	`s3tables:GetTableData` , `s3tables:GetTableM`		

Table-level activity	IAM actions		
	etadataLocation		
CREATE	s3tables:CreateTable , s3tables:UpdateTableMetadataLocation , s3tables:PutTableData , s3tables:GetTableMetadataLocation ,		
INSERT	s3tables:UpdateTableMetadataLocation , s3tables:PutTableData , s3tables:GetTableMetadataLocation		

Table-level activity	IAM actions		
UPDATE	s3tables:UpdateTableMetadataLocation , s3tables:PutTableData , s3tables:GetTableMetadataLocation		
ALTER,RENAME	s3tables:UpdateTableMetadataLocation , s3tables:PutTableData , s3tables:GetTableMetadataLocation , s3tables:RenameTable		

Table-level activity	IAM actions		
DELETE,DROP	s3tables:DeleteTable , s3tables:UpdateTableMetadataLocation , s3tables:PutTableData , s3tables:GetTableMetadataLocation		

> **Note**
>
> The `s3tables:DeleteTable` permission is required to delete a table from a table bucket. This permission allows you to permanently remove a table and all its associated data and metadata. Use this permission carefully as the delete operation cannot be undone.

Managing access to a table or database with Lake Formation

If your table buckets are integrated with the AWS analytics service using Lake Formation, then Lake Formation manages access to your tables and requires that each IAM principal (user or role) be authorized to perform actions them. Lake Formation uses its own permissions model (Lake Formation permissions) that enables fine-grained access control for Data Catalog resources.

For more information, see Overview of Lake Formation permissions in the *AWS Lake Formation Developer Guide*.

There are two main types of permissions in AWS Lake Formation:

1. Metadata access permissions control the ability to create, read, update, and delete metadata databases and tables in the Data Catalog.
2. Underlying data access permissions control the ability to read and write data to the underlying Amazon S3 locations that the Data Catalog resources point to.

Lake Formation uses a combination of its own permissions model and the IAM permissions model to control access to Data Catalog resources and underlying data:

- For a request to access Data Catalog resources or underlying data to succeed, the request must pass permission checks by both IAM and Lake Formation.
- IAM permissions control access to the Lake Formation and AWS Glue APIs and resources, whereas Lake Formation permissions control access to the Data Catalog resources, Amazon S3 locations, and the underlying data.

Lake Formation permissions apply only in the Region in which they were granted, and a principal must be authorized by a data lake administrator or another principal with the necessary permissions in order to be granted Lake Formation permissions.

Note

If you're the user who performed the table bucket integration, you already have Lake Formation permissions to your tables. If you're the only principal who will access your tables, you can skip this step. You only need to grant Lake Formation permissions on your tables to other IAM principals. This allows other principals to access the table when running queries. For more information, see Granting Lake Formation permission on a table or database.

Granting Lake Formation permission on a table or database

You can grant a principal Lake Formation permissions on a table or database in a table bucket, either through the Lake Formation console or the AWS CLI.

Note

When you grant Lake Formation permissions on a Data Catalog resource to an external account or directly to an IAM principal in another account, Lake Formation uses the

AWS Resource Access Manager (AWS RAM) service to share the resource. If the grantee account is in the same organization as the grantor account, the shared resource is available immediately to the grantee. If the grantee account is not in the same organization, AWS RAM sends an invitation to the grantee account to accept or reject the resource grant. Then, to make the shared resource available, the data lake administrator in the grantee account must use the AWS RAM console or AWS CLI to accept the invitation. For more information about cross-account data sharing, see Cross-account data sharing in Lake Formation in the *AWS Lake Formation Developer Guide*.

Console

1. Open the AWS Lake Formation console at https://console.aws.amazon.com/lakeformation/, and sign in as a data lake administrator. For more information about how to create a data lake administrator, see Create a data lake administrator in the *AWS Lake Formation Developer Guide*.
2. In the navigation pane, choose **Data permissions**, and then choose **Grant**.
3. On the **Grant Permissions** page, under **Principals**, do one of the following:
   - For Amazon Athena or Amazon Redshift, choose **IAM users and roles**, and select the IAM principal you use for queries.
   - For Amazon Data Firehose, choose **IAM users and roles**, and select the service role that you created to stream to tables.
   - For Quick, choose **SAML users and groups**, and enter the Amazon Resource Name (ARN) of your Quick admin user.
   - For AWS Glue Iceberg REST endpoint access, choose **IAM users and roles** then select the IAM role you created for you client. For more information, see Create an IAM role for your client
4. Under **LF-Tags or catalog resources**, choose **Named Data Catalog resources**.
5. For **Catalogs**, choose the subcatalog that you created when you integrated your table bucket, for example, *`account-id`*`:s3tablescatalog/`*`amzn-s3-demo-bucket`*.
6. For **Databases**, choose the S3 table bucket namespace that you created.
7. (Optional) For **Tables**, choose the S3 table that you created in your table bucket.

> **Note**
>
> If you're creating a new table in the Athena query editor, don't select a table.

8. Do one of the following:
   - If you specified a table in the prior step, for **Table permissions**, choose **Super**.
   - If you didn't specify a table in the prior step, go to **Database permissions**. For cross-account data sharing, you can't choose **Super** to grant the other principal all permissions on your database. Instead, choose more fine-grained permissions, such as **Describe**.
9. Choose **Grant**.

CLI

1. Make sure that you're running the following AWS CLI commands as a data lake administrator. For more information, see Create a data lake administrator in the *AWS Lake Formation Developer Guide*.
2. Run the following command to grant Lake Formation permissions on table in S3 table bucket to an IAM principal to access the table. To use this example, replace the *`user input placeholders`* with your own information.

```
aws lakeformation grant-permissions \
--region us-east-1 \
--cli-input-json \
'{
    "Principal": {
        "DataLakePrincipalIdentifier": "user or role ARN, for example,
 arn:aws:iam::account-id:role/example-role"
    },
    "Resource": {
        "Table": {
            "CatalogId": "account-id:s3tablescatalog/amzn-s3-demo-bucket",
            "DatabaseName": "S3 table bucket namespace, for example,
 test_namespace",
            "Name": "S3 table bucket table name, for example test_table"
        }
    },
    "Permissions": [
```

```
        "ALL"
    ]
}'
```

VPC connectivity for S3 Tables

All tables in S3 Tables are in the Apache Iceberg format and are made up of two types of S3 objects. These two types of objects are data files which store the data and metadata files which track information about the data files at different points in time. All table bucket, namespace, and table operations (for example, `CreateNamespace`, `CreateTable`, and so on) are routed through an S3 Tables endpoint (`s3tables.region.amazonaws.com`) and all object-level operations that read or write the data and metadata files continue to be routed through an S3 service endpoint (`s3.region.amazonaws.com`).

To access S3 Tables, Amazon S3 supports two types of VPC endpoints by using AWS PrivateLink: gateway endpoints and interface endpoints. A gateway endpoint is a gateway that you specify in your route table to access S3 from your VPC over the AWS network. Interface endpoints extend the functionality of gateway endpoints by using private IP addresses to route requests to Amazon S3 from within your VPC, on premises, or from a VPC in another AWS Region by using VPC peering or AWS Transit Gateway.

To access S3 Tables from a VPC, we recommend creating two VPC endpoints (one for S3 and the other for S3 Tables). You can create either a gateway or an interface endpoint to route file (object) level operations to S3 and an interface endpoint to route bucket and table-level operations to S3 Tables. You can create and use VPC endpoints for file-level requests using S3. For more information, see Gateway endpoints in the *AWS PrivateLink* User Guide.

To learn more about using AWS PrivateLink to create and work with endpoints for S3 Tables, see the following topics. To create a VPC interface endpoint, see Create a VPC endpoint in the *AWS PrivateLink Guide*.

Topics

- Creating VPC endpoints for S3 Tables
- Accessing table buckets and tables through endpoints using the AWS CLI
- Configuring a VPC network when using query engines
- Using the dual-stack endpoints to access tables and table buckets
- Using VPC endpoint policy conditions with table bucket policies

- Restricting access to S3 Tables within the VPC network

Creating VPC endpoints for S3 Tables

When you create a VPC endpoint, S3 Tables generates two types of endpoint-specific DNS names: Regional and Zonal.

- A Regional DNS name is of the following format: `VPCendpointID.s3tables.AWSregion.vpce.amazonaws.com`. For example, for the VPC endpoint ID `vpce-1a2b3c4d`, the DNS name generated will be similar to `vpce-1a2b3c4d-5e6f.s3tables.us-east-1.vpce.amazonaws.com`
- A Zonal DNS name is of the following format: `VPCendpointID-AvailabilityZone.s3tables.AWSregion.vpce.amazonaws.com`. For example, For the VPC endpoint ID `vpce-1a2b3c4d-5e6f.`, the DNS name generated will be similar to `vpce-1a2b3c4d-5e6f-us-east-1a.s3tables.us-east-1.vpce.amazonaws.com`

  A Zonal DNS name includes your Availability Zone. You might use Zonal DNS names if your architecture isolates Availability Zones. Endpoint specific S3 DNS names can be resolved from the S3 public DNS domain.

You can also use Private DNS options to simplify routing S3 traffic over VPC endpoints and help you take advantage of the lowest-cost network path available to your application. Private DNS maps the public endpoint of S3 Tables, for instance, `s3tables.region.amazonaws.com`, to a private IP in your VPC. You can use private DNS options to route Regional S3 traffic without updating your S3 clients to use the endpoint-specific DNS names of your interface endpoints.

Accessing table buckets and tables through endpoints using the AWS CLI

You can use the AWS Command Line Interface (AWS CLI) to access table buckets and tables through the interface endpoints. With the AWS CLI, `aws s3` commands route traffic through the Amazon S3 endpoint. The `aws s3tables` AWS CLI commands use the Amazon S3 Tables endpoint.

An example of an `s3tables` VPC endpoint is `vpce-0123456afghjipljw-nmopsqea.s3tables.region.vpce.amazonaws.com`

An `s3tables` VPC endpoint doesn't include a bucket name. You can access the `s3tables` VPC endpoint using the `aws s3tables` AWS CLI commands.

An example of an s3 VPC endpoint is amzn-s3-demo-bucket.vpce-0123456afghjipljw-nmopsqea.s3.region.vpce.amazonaws.com

You can access the s3 VPC endpoint using the aws s3 AWS CLI commands.

Using the AWS CLI

To access table buckets and tables through interface endpoints using the AWS CLI, use the --region- and --endpoint-url parameters. To perform table bucket and table level actions, use the S3 Tables endpoint URL. To perform object level actions, use the Amazon S3 endpoint URL.

In the following examples, replace the *user input placeholders* with your own information.

Example 1: Use an endpoint URL to list table buckets in your account

```
aws s3tables list-table-buckets --endpoint https://vpce-0123456afghjipljb-aac.s3tables.us-east-1.vpce.amazonaws.com –region us-east-1
```

Example 2: Use an endpoint URL to list tables in your bucket

```
aws s3tables list-tables --table-bucket-arn arn:aws:s3tables:us-east-1:123456789301:bucket/amzn-s3-demo-bucket --endpoint
 https://vpce-0123456afghjipljb-aac.s3tables.us-east-1.vpce.amazonaws.com --region us-east-1
```

Configuring a VPC network when using query engines

Use the following steps to configure a VPC network when using query engines.

1. To get started, you can create or update a VPC. For more information, see Create a VPC.
2. For table and table bucket level operations that route to S3 Tables, create a new interface endpoint. For more information, see Access an AWS service using an interface VPC endpoint.
3. For all object level operations that route to Amazon S3, create a gateway endpoint or a interface endpoint. For more information on gateway endpoints, see Create a gateway endpoint.
4. Next, configure your data resources and launch an Amazon EMR cluster. For more information, see Getting started with Amazon EMR.
5. You can then submit a Spark application with an additional configuration by selecting your DNS names from the VPC endpoint. For example, spark.sql.catalog.ice_catalog.s3tables.endpoint and https://interface-

endpoint.s3tables.*us-east-1*.vpce.amazonaws.com For more information, see Submit work to your Amazon EMR cluster.

Using the dual-stack endpoints to access tables and table buckets

S3 Tables supports dual-stack connectivity for AWS PrivateLink. Dual-stack endpoints allow you to access S3 tables buckets using the Internet Protocol version 6 (IPv6), in addition to the IPv4 protocol, depending on what your network supports. You can access an S3 bucket through a dual-stack endpoint using the following naming convention:

```
s3tables.<region>.api.aws
```

The following are some things you should know before trying to access S3 tables and table buckets over IPv6 in your VPC:

- The client you use to access tables and your S3 client must both have dual-stack enabled.
- IPv6 inbound is not enabled by default for VPC security groups. To allow IPv6 access you will need to add a new rule allowing HTTPS (TCP port 443) to your security group. For more information, see *Configure security group rules* in the *Amazon EC2user guide*
- If your VPC doesn't have IPv6 CIDRs assigned, you will need to manually add an IPv6 CIDR block to your VPC. For more info, see *Add IPv6 support for your VPC* in the *AWS PrivateLink user guide*
- If you use IP address filtering IAM policies they must be updated to handle IPv6 addresses. For more information about managing access permissions with IAM, see Identity and Access Management for Amazon S3.

To create a new VPC endpoint that uses the dual-stack endpoint for S3 Tables use example CLI command:

```
aws ec2 create-vpc-endpoint \
  --vpc-id vpc-id \
  --service-name com.amazonaws.aws-region.s3tables \
  --subnet-ids subnet-1 subnet-2 \
  --vpc-endpoint-type Interface \
  --ip-address-type dualstack \
  --dns-options "DnsRecordIpType=dualstack" \
  --security-group-ids sg-id \
  --region aws-region
```

For more information on creating VPC endpoints see *Create an interface endpoint* in the VPC user guide.

If your network supports IPv6 and you want to update your VPC to enable IPv6 you can use the following CLI command:

```
aws ec2 modify-vpc-endpoint \
  --vpc-endpoint-id vpce-id \
  --ip-address-type dualstack \
  --dns-options "DnsRecordIpType=dualstack" \
  --region aws-region
```

Using VPC endpoint policy conditions with table bucket policies

You can use the `aws:SourceVpce`, `aws:SourceVpc`, and `aws:VpcSourceIp` condition keys in a table bucket policy to restrict access to table bucket resources from a specific VPC or VPC endpoint. For more information about these condition keys, see AWS global condition context keys in the *IAM User Guide*.

When using the Iceberg REST APIs, S3 Tables makes requests to Amazon S3 on your behalf to read and write table metadata. When an Iceberg REST API request is made through a VPC endpoint, the source VPC of the initial request is not preserved in the table metadata requests to Amazon S3. If your table bucket policy restricts access using these condition keys, you must also add a condition for `aws:CalledVia` to allow these table metadata requests from S3 Tables.

The following example table bucket policy denies all S3 Tables actions unless the request comes from the specified VPC endpoint or is a request made by the Amazon S3 Tables Iceberg REST endpoint on your behalf.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "RestrictToVPCe",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3tables:*",
            "Resource": [
```

```
                "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket",
                "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket/*"
            ],
            "Condition": {
                "StringNotEqualsIfExists": {
                    "aws:SourceVpce": "vpce-1a2b3c4d",
                    "aws:CalledViaLast": "s3tables.amazonaws.com"
                }
            }
        }
    ]
}
```

Restricting access to S3 Tables within the VPC network

Similar to resource-based policies, you can attach an endpoint policy to your VPC endpoint that controls the access to tables and table buckets. In the following example, the interface endpoint policy restricts access to only specific table buckets.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "Policy141511512309",
    "Statement": [
        {
            "Sid": "Access-to-specific-bucket-only",
            "Principal": "*",
            "Action": "s3tables:*",
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket",
                "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket/*"
            ]
        }
    ]
}
```

Security considerations and limitations for S3 Tables

The following list describes which security and access control features and functionality are unsupported or limited for S3 Tables.

- Public access policies are not supported. Users can't modify bucket or table policies to allow public access.
- Presigned URLs to access objects associated with a table are not supported.
- Requests made over HTTP are not supported. Amazon S3 automatically responds with an HTTP redirect for any requests made over HTTP to upgrade the requests to HTTPS.
- You must use AWS Signature Version 4 when making requests to an access point by using the REST APIs.
- Requests made over the Internet Protocol version 6 (IPv6) are supported only for object-level actions over table storage endpoints, and not for the table- and bucket-level actions.
- Table bucket and table access policies are limited to 20 KB in size.

Logging and monitoring for S3 Tables

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon S3 Tables. We recommend collecting monitoring data from your tables in table buckets so that you can more easily debug a multipoint failure if one occurs.

AWS provides several tools for monitoring your S3 Tables resources and responding to potential incidents.

AWS CloudTrail Logs

CloudTrail provides a record of actions taken by a user, role, or an AWS service in Amazon S3. Using the information collected by CloudTrail, you can determine the request that was made to Amazon S3, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging Amazon S3 API calls using AWS CloudTrail.

Amazon S3 Access Logs

Server access logs provide detailed records about the requests that are made to a table bucket. When you enable server access logging on a table bucket, Amazon S3 delivers a log record for

every request to that table bucket to a general purpose bucket that you specify. These logs are useful for security and access auditing. For more information, see Logging requests for S3 Tables using server access logging.

Amazon CloudWatch Alarms

Using Amazon CloudWatch alarms, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon SNS topic or AWS Auto Scaling policy. CloudWatch alarms do not invoke actions because they are in a particular state. Rather, the state must have changed and been maintained for a specified number of periods. For more information, see Monitoring metrics with Amazon CloudWatch.

Topics

- Logging with AWS CloudTrail for S3 Tables
- Logging requests for S3 Tables using server access logging
- Monitoring metrics with Amazon CloudWatch

Logging with AWS CloudTrail for S3 Tables

Amazon S3 is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures all API calls for Amazon S3 as events. Using the information collected by CloudTrail, you can determine the request that was made to Amazon S3, the IP address from which the request was made, when it was made, and additional details. When a supported event activity occurs in Amazon S3, that activity is recorded in a CloudTrail event. You can use AWS CloudTrail trail to log management events and data events for S3 Tables. For more information, see Amazon S3 CloudTrail events and What is AWS CloudTrail? in the *AWS CloudTrailUser Guide*.

CloudTrail management events for S3 Tables

Management events provide information about management operations that are performed on resources in your AWS account.

By default, CloudTrail logs management events for S3 Tables. The `eventsource` for CloudTrail management events for S3 Tables is `s3tables.amazonaws.com`. When you set up your AWS account, CloudTrail management events are enabled by default. The following API actions are tracked by CloudTrail and logged as management events.

- CreateNamespace
- CreateTable
- CreateTableBucket
- DeleteNamespace
- DeleteTable
- DeleteTableBucket
- DeleteTableBucketPolicy
- DeleteTablePolicy
- GetNamespace
- GetTable
- GetTableBucket
- GetTableBucketMaintenanceConfiguration
- GetTableBucketPolicy
- GetTableMaintenanceConfiguration
- GetTableMaintenanceJobStatus
- GetTableMetadataLocation
- GetTablePolicy
- ListNamespaces
- ListTableBuckets
- ListTables
- PutTableBucketMaintenanceConfiguration
- PutTableMaintenanceConfiguration
- PutBucketPolicy
- PutTablePolicy
- RenameTable
- UpdateTableMetadataLocation

For more information on CloudTrail management events, see Logging management events in the *AWS CloudTrail User Guide*.

CloudTrail management events for S3 Tables maintenance

S3 logs automatic maintenance operations as `TablesMaintenanceEvent` management events in CloudTrail. These events occur during operations like compaction and snapshot expiration. For more information about S3 table maintenance, see Maintenance for tables.

How to identify maintenance events

You can identify S3 Tables maintenance events in CloudTrail logs by these attribute values:

- `eventSource: s3tables.amazonaws.com`
- `eventType: AwsServiceEvent`
- `eventName: TablesMaintenanceEvent`
- `userAgent: maintenance.s3tables.amazonaws.com`
- `activityType:`
  - `IcebergCompaction` (for compaction)
  - `IcebergSnapshotManagement` (for snapshot expiration)

For an example of a compaction maintenance event, see Example – CloudTrail log file for a table maintenance management event.

CloudTrail data events for S3 Tables

Data events provide information about the resource operations performed on or in a resource.By default, CloudTrail trails don't log data events, but you can configure trails to log data events.

When you log data events for a trail in CloudTrail, you will choose or specify the resource type. S3 Tables has two resources types, `AWS::S3Tables::Table` and `AWS::S3Tables::TableBucket`.

The following data events are logged to CloudTrail.

- `AbortMultipartUpload`
- `CompleteMultipartUpload`
- `CreateMultipartUpload`
- `GetObject`
- `HeadObject`
- `ListParts`

- `PutObject`
- `UploadPart`

For more information on CloudTrail data events, see Logging data events in the *AWS CloudTrail User Guide*.

For additional information about CloudTrail events for S3 Tables, see the following topics:

Topics

- AWS CloudTrail data event log file examples for S3 Tables

AWS CloudTrail data event log file examples for S3 Tables

A AWS CloudTrail log file includes information about the requested API operation, the date and time of the operation, request parameters, and so on. This topic provides example log files for CloudTrail data events for S3 Tables.

Topics

- Example – CloudTrail log file for GetObject data event
- Example – CloudTrail log file for a table maintenance management event
- Example – CloudTrail log file for PutObject data event

Example – CloudTrail log file for `GetObject` data event

The following example shows a CloudTrail log file example that demonstrates the `GetObject` API operation.

```
    {
        "eventVersion": "1.11",
        "userIdentity": {
          "type": "IAMUser",
          "principalId": "123456789012",
          "arn": "arn:aws:iam::111122223333:user/"myUserName",
          "accountId": "111122223333",
          "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
          "userName":"myUserName"
        },
        "eventTime": "2024-11-22T17:12:25Z",
```

```
        "eventSource": "s3tables.amazonaws.com",
        "eventName": "GetObject",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "[aws-cli/2.18.5]",
        "requestParameters": {
            "Host": "tableWarehouseLocation.s3.us-east-1.amazonaws.com",
            "key": "product-info.json"
        },
        "responseElements":  null,
        "additionalEventData": {
            "SignatureVersion": "SigV4",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "bytesTransferredIn": 0,
            "AuthenticationMethod": "AuthHeader",
            "xAmzId2": "q6xhNJYmhg",
            "bytesTransferredOut": 28441

          },
          "requestID": "07D681123BD12AED",
          "eventID": "f2b287f3-0df1-1234-a2f4-c4bdfed47657",
          "readOnly": true,
          "resources": [{
              "accountId": "111122223333",
              "type": "AWS::S3Tables::TableBucket",
               "ARN": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket1"
           }, {
              "accountId": "111122223333",
              "type": "AWS::S3Tables::Table",
              "ARN": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
bucket/table/111aa1111-22bb-33cc-44dd-5555eee66ffff"

           }],
           "eventType": "AwsApiCall",
           "managementEvent": false,
           "recipientAccountId": "444455556666",
           "eventCategory": "Data",
           "tlsDetails": {
             "tlsVersion": "TLSv1.2",
             "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
             "clientProvidedHostHeader": "tableWarehouseLocation.s3.us-
east-1.amazonaws.com"
          }
```

```
    }
```

Example – CloudTrail log file for a table maintenance management event

The following is a CloudTrail log file example that demonstrates a maintenance event for table compaction performed by S3 as part of automatic table maintenance. For more information on events for table maintenance, see CloudTrail management events for S3 Tables maintenance

```
{
  "eventVersion": "1.11",
  "userIdentity": {
    "type": "AWSService",
    "invokedBy": "maintenance.s3tables.amazonaws.com"
  },
  "eventTime": "2025-09-18T20:13:14Z",
  "eventSource": "s3tables.amazonaws.com",
  "eventName": "TablesMaintenanceEvent",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "maintenance.s3tables.amazonaws.com",
  "userAgent": "maintenance.s3tables.amazonaws.com",
  "requestParameters": null,
  "responseElements": null,
  "eventID": "b8f96329-ef5c-32b5-94f6-eeed9061ea32",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::S3Tables::TableBucket",
      "ARN": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket"
    },
    {
      "accountId": "111122223333",
      "type": "AWS::S3Tables::Table",
      "ARN": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket/
table/7ff7750e-23b3-481e-a90a-7d87d423d336"
    }
  ],
  "eventType": "AwsServiceEvent",
  "managementEvent": true,
  "recipientAccountId": "111122223333",
  "sharedEventID": "62a57826-a66e-479b-befa-0e65663ee9e8",
  "serviceEventDetails": {
    "activityType": "icebergCompaction"
```

```
  },
  "eventCategory": "Management"
}
```

Example – CloudTrail log file for `PutObject` data event

The following example shows a CloudTrail log file example that demonstrates the `PutObject` API operation.

```
{
        "eventVersion": "1.11",
        "userIdentity": {
          "type": "IAMUser",
          "principalId": "123456789012",
          "arn":  "arn:aws:iam::444455556666:user/"myUserName",
          "accountId": "444455556666",
          "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
          "userName":"myUserName"
        },
        "eventTime": "2024-11-22T17:12:25Z",
        "eventSource": "s3tables.amazonaws.com",
        "eventName": "PutObject",
        "awsRegion": "us-east-1",
        "sourceIPAddress": "192.0.2.0",
        "userAgent": "[aws-cli/2.18.5]",
        "requestParameters": {
            "Host": "tableWarehouseLocation.s3.us-east-1.amazonaws.com",
            "key": "product-info.json"
        },
        "responseElements":  {
            "x-amz-server-side-encryption": "AES256",
            "x-amz-version-id": "13zAFMdccAjt3MWd6ehxgCCCDRdkAKDw"
        },
        "additionalEventData": {
            "SignatureVersion": "SigV4",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "bytesTransferredIn": 28441,
            "AuthenticationMethod": "AuthHeader",
            "xAmzId2": "q6xhCJYmhg",
            "bytesTransferredOut": 0

          },
```

```
          "requestID": "28d2faaf-1234-4649-997d-EXAMPLE72818",
          "eventID": "694d604a-d190-1234-0dd1-EXAMPLEe20c1",
          "readOnly": false,
          "resources": [{
              "accountId": "444455556666",
              "type": "AWS::S3Tables::TableBucket",
               "ARN": "arn:aws:s3tables:us-east-1:444455556666:bucket/amzn-s3-demo-
bucket1"
           }, {
              "accountId": "444455556666",
              "type": "AWS::S3Tables::Table",
              "ARN": "arn:aws:s3tables:us-east-1:444455556666:bucket/amzn-s3-demo-
bucket1/table/b89ec883-b1d9-4b37-9cd7-b86f590123f4"
           }],
           "eventType": "AwsApiCall",
           "managementEvent": false,
           "recipientAccountId": "111122223333",
           "eventCategory": "Data",
           "tlsDetails": {
             "tlsVersion": "TLSv1.2",
             "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
             "clientProvidedHostHeader": "tableWarehouseLocation.s3.us-
east-1.amazonaws.com"
            }
          }
```

Logging requests for S3 Tables using server access logging

Server access logging for S3 Tables provides detailed records for the requests that are made to a table bucket. When you enable server access logging on a table bucket, Amazon S3 generates a log record for every request that is made to that table bucket, and delivers the log files to a general purpose bucket that you specify. These logs can help you with security and access auditing, access analysis, and cost attribution.

Server access logging is configured at the table bucket level. By default, Amazon S3 doesn't collect server access logs for table buckets. When you enable logging on a table bucket, Amazon S3 logs requests for all namespaces and tables in that table bucket.

This topic describes server access logging for S3 Tables. If you want to enable server access logging for general purpose buckets instead, see Logging requests with server access logging.

How server access logging works for S3 Tables

You turn server access logging on or off at the table bucket level. To enable or update logging, you use the `PutTableBucketLogging` operation with a `tableBucketLoggingStatus` that contains a `loggingEnabled` element. The `loggingEnabled` element specifies the following values:

- `targetBucket` – Required. The name of the destination general purpose bucket where Amazon S3 delivers the log files.
- `targetPrefix` – Required. The key prefix that Amazon S3 uses for the delivered log objects, for example `logs/`. You can specify an empty prefix. Amazon S3 doesn't add a slash (`/`) after the prefix. For more information, see Log delivery location and key format.
- `targetObjectKeyFormat` – Required. Contains a `partitionedPrefix` element with a `partitionDateSource` value that controls the date that Amazon S3 uses to partition the delivered log objects. The value is either `EventTime` or `DeliveryTime`.

To view the current logging configuration for a table bucket, use the `GetTableBucketLogging` operation. After you enable logging, it might take a few hours before Amazon S3 begins delivering log files to the destination general purpose bucket.

Note

The following limitations apply to server access logging for S3 Tables:

- Server access logging for S3 Tables is available in all AWS Regions.
- Logging is configured at the table bucket level only. When you enable logging, all namespaces and tables in the table bucket are logged. There is no per-namespace or per-table setting.
- Delivery to Amazon CloudWatch Logs is not supported. Amazon S3 delivers S3 Tables server access logs only to a general purpose bucket.
- You can't enable server access logging on AWS managed tables.
- Server access logging for S3 Tables doesn't support the `SimplePrefix` log object key format. For more information, see SimplePrefix.
- Server access logging for S3 Tables doesn't support ACLs.

- Server access logging for S3 Tables doesn't log requests for table buckets that are made to the wrong Regional endpoint. S3 Tables only supports requests that are made to the correct Regional endpoint.

Best-effort log delivery

S3 Tables server access log records are delivered on a best-effort basis. Most requests for a table bucket that is properly configured for logging result in a delivered log record. Most log records are delivered within a few hours of the time that they are recorded, but they can be delivered more frequently.

The completeness and timeliness of server logging is not guaranteed. The log record for a particular request might be delivered long after the request was actually processed, or *it might not be delivered at all*. You might even see a duplicate log record. The purpose of server logs is to give you an idea of the nature of traffic against your table bucket. Although log records are rarely lost or duplicated, be aware that server logging is not meant to be a complete accounting of all requests.

For guaranteed and more complete audit logging, use AWS CloudTrail. CloudTrail captures all API calls for S3 Tables as events. For more information, see Logging with AWS CloudTrail for S3 Tables.

Logging status changes take effect over time

Changes to the logging status of a table bucket take time to actually affect the delivery of log files. For example, if you enable logging for a table bucket, some requests made in the following hour might be logged, and others might not. Suppose that you change the destination bucket for logging from bucket A to bucket B. For the next hour, some logs might continue to be delivered to bucket A, whereas others might be delivered to the new destination bucket B. In all cases, the new settings eventually take effect without any further action on your part.

Destination general purpose bucket requirements

Before you enable server access logging, confirm that the destination general purpose bucket meets the following requirements:

- The destination general purpose bucket must be in the same AWS account and the same AWS Region as the source table bucket. Cross-account and cross-Region delivery are not supported.
- The destination general purpose bucket must use Amazon S3 managed keys (SSE-S3) for default encryption. You can't use server-side encryption with AWS KMS keys (SSE-KMS) or dual-layer

server-side encryption with AWS KMS keys (DSSE-KMS). If the destination general purpose bucket uses SSE-KMS or DSSE-KMS, Amazon S3 might deliver log objects that are encrypted with a key that you can't access.

- The destination general purpose bucket must not have S3 Object Lock enabled.
- The destination general purpose bucket must not have Requester Pays enabled.

For simpler log management, we recommend that you deliver logs to a general purpose bucket that you use only for server access logs.

Permissions for server access logging

Server access logging for S3 Tables requires two kinds of permissions: a bucket policy on the destination general purpose bucket that lets Amazon S3 deliver log files, and AWS Identity and Access Management permissions for the user or role that configures logging.

Destination bucket policy

Amazon S3 delivers log files by using the logging service principal (`logging.s3.amazonaws.com`). You must grant this service principal the `s3:PutObject` permission on the destination general purpose bucket. This uses the service principal model rather than the legacy log delivery access control list (ACL) group.

If you use the Amazon S3 console to enable server access logging, the console automatically updates the bucket policy on the destination general purpose bucket to grant this permission. If you enable logging with the AWS SDKs, the AWS Command Line Interface (AWS CLI), or the Amazon S3 REST API, you must add the bucket policy to the destination general purpose bucket yourself.

The following example bucket policy grants the `s3:PutObject` permission to the logging service principal. To use this bucket policy, replace the *`user input placeholders`* with your own information.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3TablesServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
```

```
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-logging-bucket/logs/*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3tables:us-
east-1:111122223333:bucket/amzn-s3-demo-table-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

> **Note**
>
> If there are Deny statements in your bucket policy, make sure that they don't prevent Amazon S3 from delivering access logs.

Permissions to configure logging

To configure server access logging, the user or role that makes the request must have permission to call the `s3tables:PutTableBucketLogging` and `s3tables:GetTableBucketLogging` actions on the table bucket. The following example AWS Identity and Access Management policy grants these permissions. To use this policy, replace the *`user input placeholders`* with your own information.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ConfigureS3TablesServerAccessLogging",
            "Effect": "Allow",
            "Action": [
                "s3tables:PutTableBucketLogging",
                "s3tables:GetTableBucketLogging"
            ],
```

```
            "Resource": "arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-
table-bucket"
        }
    ]
}
```

Enabling or updating server access logging

You can enable or update server access logging for a table bucket by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. In the **Table buckets** list, choose the name of the table bucket that you want to enable server access logging for.
4. Choose **Properties**.
5. In the **Server access logging** section, choose **Edit**.
6. Under **Server access logging**, choose **Enable**.
7. Under **Destination bucket**, specify a general purpose bucket and an optional prefix. If you specify a prefix, we recommend including a forward slash (/) after the prefix to make it easier to find your logs.

> **Note**
>
> Specifying a prefix with a slash (/) makes it simpler for you to locate the log objects. For example, if you specify the prefix value `logs/`, each log object that Amazon S3 creates begins with the `logs/` prefix in its key, as follows:
>
> ```
> logs/2013-11-01-21-32-16-E568B2907131C0C0
> ```
>
> If you specify the prefix value `logs` without a slash, the log object appears as follows:
>
> ```
> logs2013-11-01-21-32-16-E568B2907131C0C0
> ```

8. Under **Source of date used in log object key format**, do one of the following:

   - Choose **S3 event time** so that the year, month, and day in the log object key are based on the time of the S3 event in the delivered file. This option corresponds to the `EventTime` value for `partitionDateSource` in the `partitionedPrefix` element of `targetObjectKeyFormat`, and is the default.
   - Choose **Log file delivery time** so that the year, month, and day in the log object key are based on when the log file was delivered to Amazon S3. This option corresponds to the `DeliveryTime` value for `partitionDateSource` in the `partitionedPrefix` element of `targetObjectKeyFormat`.

9. Choose **Save changes**.

   When you enable server access logging on a table bucket, the console enables logging on the source table bucket. The console also updates the bucket policy for the destination general purpose bucket to grant the `s3:PutObject` permission to the logging service principal (`logging.s3.amazonaws.com`). For more information about this bucket policy, see Destination bucket policy.

   You can view the logs in the destination general purpose bucket. After you enable server access logging, it might take a few hours before the logs are delivered.

Using the AWS CLI

Enable or update logging

To enable or update server access logging, use the `put-table-bucket-logging` command. The following example enables logging on the table bucket `amzn-s3-demo-table-bucket`. It delivers logs to the general purpose bucket `amzn-s3-demo-logging-bucket` under the `logs/` prefix and partitions the log object keys by event time. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3tables put-table-bucket-logging \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket \
    --table-bucket-logging-status '{
        "loggingEnabled": {
            "targetBucket": "amzn-s3-demo-logging-bucket",
            "targetPrefix": "logs/",
            "targetObjectKeyFormat": {
```

```
                "partitionedPrefix": {
                    "partitionDateSource": "EventTime"
                }
            }
        }
    }'
```

To partition log object keys by the time that the log file was delivered, specify `DeliveryTime` for `partitionDateSource`. The `targetObjectKeyFormat` value is required.

View the logging configuration

To view the current logging configuration for a table bucket, use the `get-table-bucket-logging` command.

```
aws s3tables get-table-bucket-logging \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket
```

When logging is enabled, the output looks like the following:

```
{
    "tableBucketLoggingStatus": {
        "loggingEnabled": {
            "targetBucket": "amzn-s3-demo-logging-bucket",
            "targetPrefix": "logs/",
            "targetObjectKeyFormat": {
                "partitionedPrefix": {
                    "partitionDateSource": "EventTime"
                }
            }
        }
    }
}
```

When logging is disabled, the output contains an empty logging status:

```
{
    "tableBucketLoggingStatus": {}
}
```

Disable logging

To disable server access logging, use the `put-table-bucket-logging` command with an empty logging status.

```
aws s3tables put-table-bucket-logging \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket \
    --table-bucket-logging-status '{}'
```

Using the AWS SDKs

The following example uses the AWS SDK for Python (Boto3) to enable server access logging on a table bucket, view the configuration, and then disable logging. Replace the *user input placeholders* with your own information.

```
import boto3

client = boto3.client('s3tables')

table_bucket_arn = 'arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket'

# Enable or update server access logging.
client.put_table_bucket_logging(
    tableBucketARN=table_bucket_arn,
    tableBucketLoggingStatus={
        'loggingEnabled': {
            'targetBucket': 'amzn-s3-demo-logging-bucket',
            'targetPrefix': 'logs/',
            'targetObjectKeyFormat': {
                'partitionedPrefix': {
                    'partitionDateSource': 'EventTime'
                }
            }
        }
    }
)

# View the current logging configuration.
response = client.get_table_bucket_logging(tableBucketARN=table_bucket_arn)
print(response['tableBucketLoggingStatus'])

# Disable server access logging.
client.put_table_bucket_logging(
```

```
    tableBucketARN=table_bucket_arn,
    tableBucketLoggingStatus={}
)
```

Using the REST API

Enable or update logging

To enable or update logging, submit a `PutTableBucketLogging` request. The table bucket ARN is URL-encoded in the request path, and the request body contains the logging status. A successful request returns `204 No Content`.

```
PUT /buckets/arn%3Aaws%3As3tables%3Aus-east-1%3A111122223333%3Abucket%2Famzn-s3-demo-table-bucket/logging HTTP/1.1
Host: s3tables.us-east-1.amazonaws.com
Content-Type: application/json

{
    "tableBucketLoggingStatus": {
        "loggingEnabled": {
            "targetBucket": "amzn-s3-demo-logging-bucket",
            "targetPrefix": "logs/",
            "targetObjectKeyFormat": {
                "partitionedPrefix": {
                    "partitionDateSource": "EventTime"
                }
            }
        }
    }
}
```

View the logging configuration

To retrieve the current logging configuration, submit a `GetTableBucketLogging` request.

```
GET /buckets/arn%3Aaws%3As3tables%3Aus-east-1%3A111122223333%3Abucket%2Famzn-s3-demo-table-bucket/logging HTTP/1.1
Host: s3tables.us-east-1.amazonaws.com
```

Disable logging

To disable logging, submit a `PutTableBucketLogging` request with an empty logging status.

```
PUT /buckets/arn%3Aaws%3As3tables%3Aus-east-1%3A111122223333%3Abucket%2Famzn-s3-demo-table-bucket/logging HTTP/1.1
Host: s3tables.us-east-1.amazonaws.com
Content-Type: application/json

{
    "tableBucketLoggingStatus": {}
}
```

Disabling server access logging

To disable server access logging, send a `PutTableBucketLogging` request with an empty logging status. There is no separate operation to delete the logging configuration. With the AWS CLI, run the `put-table-bucket-logging` command with `--table-bucket-logging-status '{}'`. With Boto3, call `put_table_bucket_logging` with `tableBucketLoggingStatus={}`.

```
aws s3tables put-table-bucket-logging \
    --table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket \
    --table-bucket-logging-status '{}'
```

To disable logging by using the Amazon S3 console, choose the table bucket, choose **Properties**, and in the **Server access logging** section choose **Edit**. Then choose **Disable**, and choose **Save changes**.

When you disable logging, Amazon S3 stops generating log records for the table bucket. Log files that were already delivered to the destination general purpose bucket are not affected.

Log delivery location and key format

Amazon S3 delivers log files to the destination general purpose bucket by using a key that identifies the source table bucket. Log objects use the following key format:

```
s3://targetBucket/targetPrefixaccountId/region/tableBucketName/YYYY/MM/DD/YYYY-MM-DD-hh-mm-ss-UniqueString
```

The following example shows a delivered log object key:

```
s3://amzn-s3-demo-logging-bucket/logs/111122223333/us-east-1/amzn-s3-demo-table-bucket/2026/10/28/2026-10-28-00-00-00-4520D3E0684DC910
```

The key format uses the following parts:

- *targetPrefix* – Amazon S3 appends the prefix exactly as you provide it and doesn't insert a slash (/) after it. A prefix of `logs` without a trailing slash produces a key that begins `logs111122223333/`. To keep log objects organized, end the prefix with a slash (/). If you leave the prefix empty, the key begins with the account ID.
- *accountId* – The AWS account that owns the source table bucket.
- *region* – The AWS Region of the source table bucket.
- *tableBucketName* – The name of the source table bucket. All records for the table bucket, its namespaces, its tables, and data requests are delivered under this single folder. There are no per-table or per-namespace folders.
- *YYYY*/*MM*/*DD*/ – The date folders. S3 Tables server access logs are always date-partitioned.

The `partitionDateSource` value in the `partitionedPrefix` element of `targetObjectKeyFormat` controls how Amazon S3 chooses the date:

- `EventTime` (default) – The date folders are based on the date of the request. The time in the file name is always `00-00-00`.
- `DeliveryTime` – The date folders are based on the date that the log file was delivered. The time in the file name is the actual delivery time.

Server access log format for S3 Tables

Server access logs for S3 Tables are delivered in space-delimited text format. Each log record represents one request and contains 25 fields in the following order:

```
table_bucket_arn [request_time] table_arn namespace_name table_name key object_size
 remote_ip requester request_id operation http_status error_code bytes_sent
 total_time turnaround_time "referer" "user_agent" signature_version cipher_suite
 authentication_type host_header tls_version request_time_iso source_region
```

In a log record, the `request_time` field is enclosed in square brackets, and the `referer` and `user_agent` fields are enclosed in double quotation marks. A hyphen (-) indicates a field that doesn't apply to the request.

The following table describes each field.

Field	Description	Example
table_bucket_arn	The ARN of the table bucket that received the request.	arn:aws:s3tables:us-east-2:111122223333:bucket/amzn-s3-demo-table-bucket
request_time	The time that the request was received, enclosed in square brackets.	[28/Oct/2026:13:46:39 +0000]
table_arn	The ARN of the table, or - for table bucket and namespace requests.	arn:aws:s3tables:us-east-2:111122223333:bucket/amzn-s3-demo-table-bucket/table/cdae79fb-03e3-49dc-92c7-10e279e37e8f
namespace_name	The namespace in the request, or -.	sal_ns
table_name	The table name in the request, or -.	events
key	The object key for data requests, or -.	metadata/00000-a3001105-metadata.json
object_size	The object size in bytes for data requests, or -.	756
remote_ip	The IP address of the requester.	15.248.6.220
requester	The IAM identity of the requester.	arn:aws:sts::111122223333:assumed-role/Admin/example-session
request_id	The request ID.	6TBZ2YZN2PGW74S8
operation	The operation that was performed.	S3Tables.GetTable

Field	Description	Example
`http_status`	The HTTP status code of the response.	`200`
`error_code`	The error code, or `-` if there was no error.	`NotFoundException`
`bytes_sent`	The number of response bytes sent, or `-`.	`735`
`total_time`	The total time to serve the request, in milliseconds.	`49`
`turnaround_time`	The time that Amazon S3 spent processing the request, in milliseconds.	`49`
`referer`	The value of the HTTP Referer header, enclosed in double quotation marks.	`"https://analytics.example.com/"`
`user_agent`	The value of the HTTP User-Agent header, enclosed in double quotation marks.	`"aws-cli/2.34.34"`
`signature_version`	The signature version that was used to authenticate the request.	`SigV4`
`cipher_suite`	The TLS cipher suite that was used for the request.	`TLS_AES_128_GCM_SHA256`
`authentication_type`	The authentication type: `AuthHeader` for an authorization header, or `QueryString` for a presigned URL.	`AuthHeader`
`host_header`	The endpoint that was used for the request.	`s3tables.us-east-2.amazonaws.com`

Field	Description	Example
`tls_version`	The TLS version that was used for the request.	`TLSv1.3`
`request_time_iso`	The time that the request was received, in ISO 8601 format with milliseconds.	`2026-10-28T13:46:39.000Z`
`source_region`	The AWS Region that the request came from, or `-`.	`us-west-2`

Note

The `operation` field is declared as `S3Tables.`*`ActionName`* for S3 Tables API requests and Iceberg REST catalog endpoint requests, or `REST.`*`HTTP_method`*`.`*`resource_type`* for requests to table storage. For Iceberg REST catalog endpoint requests, *`ActionName`* is the IAM action that the request is authorized as. For example, a `loadTable` request is logged as `S3Tables.GetTableMetadataLocation`.

The following example shows a log record for a `GetTable` request.

```
arn:aws:s3tables:us-east-2:111122223333:bucket/amzn-s3-demo-table-bucket [28/
Oct/2026:13:46:35 +0000] arn:aws:s3tables:us-east-2:111122223333:bucket/amzn-s3-
demo-table-bucket/table/cdae79fb-03e3-49dc-92c7-10e279e37e8f sal_ns events - -
 15.248.6.88 arn:aws:sts::111122223333:assumed-role/Admin/example-session c4e215be-
ad2b-4b55-a190-4225aada1885 S3Tables.GetTable 200 - 735 49 49 "-" "aws-cli/2.34.34"
 SigV4 TLS_AES_128_GCM_SHA256 AuthHeader s3tables.us-east-2.amazonaws.com TLSv1.3
 2026-10-28T13:46:35.634Z us-west-2
```

Managing log retention

A destination general purpose bucket that receives server access logs can accumulate many log objects over time. You can use Amazon S3 Lifecycle configuration to set rules so that Amazon S3 automatically deletes log objects after a specified period. If you specified a prefix in your logging configuration, you can scope the lifecycle rule to that prefix. For example, if your log objects

have the prefix `logs/`, you can create a lifecycle rule to delete all objects with that prefix after a specified number of days. For more information, see Managing the lifecycle of objects.

Keep the following behavior in mind when you manage log retention:

- If you delete a table or a table bucket, the log files that were already delivered to the destination general purpose bucket persist. They are managed independently of the source table bucket.
- If you delete the destination general purpose bucket, the logging configuration remains on the table bucket, but Amazon S3 silently drops log records until you recreate the destination general purpose bucket or update the logging configuration.

Charges for server access logging

There is no additional S3 Tables charge for generating and delivering server access logs. You pay the standard Amazon S3 charges for the destination general purpose bucket, including storage of the delivered log files, requests, and data transfer for accessing the log files. You can delete the log files at any time. For current rates, see Amazon S3 pricing.

Querying server access logs by using Amazon Athena

You can analyze S3 Tables server access logs by using Amazon Athena. The following example shows how to create a database and table in Athena, and then query the delivered log files. Replace the *`user input placeholders`* with your own information. For more information about using Athena with S3 Tables, see Querying Amazon S3 tables with Athena.

1. Open the Athena console at https://console.aws.amazon.com/athena/.
2. In the query editor, run the following command to create a database. Replace *`s3_tables_access_logs_db`* with the name that you want to give to your database.

```
CREATE DATABASE s3_tables_access_logs_db
```

3. Run the following command to create a table schema in the database. For `LOCATION`, enter the destination general purpose bucket, prefix, account ID, Region, and source table bucket name. Amazon S3 appends the prefix without inserting a slash, so there is no slash between *`targetPrefix`* and the account ID.

```
CREATE EXTERNAL TABLE s3_tables_access_logs_db.s3_tables_logs(
  `table_bucket_arn` STRING,
  `request_time` STRING,
```

```
  `table_arn` STRING,
  `namespace_name` STRING,
  `table_name` STRING,
  `key` STRING,
  `object_size` BIGINT,
  `remote_ip` STRING,
  `requester` STRING,
  `request_id` STRING,
  `operation` STRING,
  `http_status` STRING,
  `error_code` STRING,
  `bytes_sent` BIGINT,
  `total_time` STRING,
  `turnaround_time` STRING,
  `referer` STRING,
  `user_agent` STRING,
  `signature_version` STRING,
  `cipher_suite` STRING,
  `authentication_type` STRING,
  `host_header` STRING,
  `tls_version` STRING,
  `request_time_iso` STRING,
  `source_region` STRING)
PARTITIONED BY (
  `timestamp` string)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  'input.regex'='([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) (\"[^\"]*
\"|-) (\"[^\"]*\"|-) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*).*$')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://amzn-s3-demo-logging-bucket/logs/111122223333/us-east-1/amzn-s3-demo-table-
bucket/'
TBLPROPERTIES (
  'projection.enabled'='true',
  'projection.timestamp.format'='yyyy/MM/dd',
  'projection.timestamp.interval'='1',
  'projection.timestamp.interval.unit'='DAYS',
  'projection.timestamp.range'='2026/01/01,NOW',
```

```
  'projection.timestamp.type'='date',
  'storage.location.template'='s3://amzn-s3-demo-logging-
bucket/logs/111122223333/us-east-1/amzn-s3-demo-table-bucket/${timestamp}')
```

4. In the navigation pane, under **Database**, choose your database.
5. Under **Tables**, choose **Preview table** next to your table name. In the **Results** pane, you see data from the server access logs. You can now query the S3 Tables server access logs.

Monitoring metrics with Amazon CloudWatch

You can use Amazon CloudWatch metrics to track performance, detect anomalies, and monitor the operational health of tables. There are several sets of CloudWatch metrics that you can use with S3 Tables.

Daily storage metrics for tables and table buckets

Monitor the amount of data stored in tables and table buckets, including total size in bytes and number of files. These metrics track total storage bytes per access tier and file counts at the table bucket, table, and namespace level. Storage metrics for S3 Tables are reported once per day and are provided to all customers at no additional cost.

Table maintenance metrics

Monitor automated maintenance operations performed by Amazon S3 on your tables, such as compaction. These metrics track the number of bytes and files processed during maintenance activities. Maintenance metrics for S3 Tables are reported once per day and are provided to all customers at no additional cost.

Request metrics

Monitor S3 Tables requests to quickly identify and act on operational issues. These CloudWatch metrics can be optionally enabled for individual table buckets. Request metrics for S3 Tables are reported once every minute and are billed at the same rate as CloudWatch custom metrics. Request metrics include:

- counts of data plane operations (GET, PUT, HEAD, POST)
- bytes transferred
- latency measurements
- error rates

Note

Best-effort CloudWatch metrics delivery
CloudWatch metrics are delivered on a best-effort basis. Most requests for an Amazon S3 object that have request metrics result in a data point being sent to CloudWatch.
The completeness and timeliness of metrics are not guaranteed. The data point for a particular request might be returned with a timestamp that is later than when the request was actually processed. The data point for a minute might be delayed before being available through CloudWatch, or it might not be delivered at all. CloudWatch request metrics give you an idea of the nature of traffic against your bucket in near-real time.
It is not meant to be a complete accounting of all requests. It follows from the best-effort nature of this feature that the reports available at the Billing & Cost Management Dashboard might include one or more access requests that do not appear in the bucket metrics.

Metrics and dimensions

The storage metrics and dimensions that S3 Tables sends to Amazon CloudWatch are listed in the following tables.

Note

Best-effort CloudWatch metrics delivery
CloudWatch metrics are delivered on a best-effort basis. Most requests for an Amazon S3 object that have request metrics result in a data point being sent to CloudWatch.
The completeness and timeliness of metrics are not guaranteed. The data point for a particular request might be returned with a timestamp that is later than when the request was actually processed. The data point for a minute might be delayed before being available through CloudWatch, or it might not be delivered at all. CloudWatch request metrics give you an idea of the nature of traffic against your bucket in near-real time.
It is not meant to be a complete accounting of all requests. It follows from the best-effort nature of this feature that the reports available at the Billing & Cost Management Dashboard might include one or more access requests that do not appear in the bucket metrics.

Daily storage metrics for table buckets in CloudWatch

The `AWS/S3/Tables` namespace includes the following daily storage metrics which are available at no additional cost. You can filter these metrics by table bucket, table, or namespace name.

Daily storage metrics

Metric Name	Description	Units	Statistics	Granularity
Total Bucket Storage	The amount of storage in bytes used by all tables in a table bucket	Bytes	Sum	Daily
Total number of files	The total count of all files stored in a table bucket	Count	Sum	Daily

Table maintenance metrics

The `AWS/S3/Tables` namespace includes the following table maintenance metrics which are available at no additional cost. You can filter these metrics by table bucket, table, or namespace name.

Table maintenance metrics

Metric Name	Description	Units	Statistics	Granularity
CompactionBytesProcessed	The number of bytes processed during table compaction operations	Bytes	Sum	Daily
CompactionObjectsCount	The number of objects processed during table compaction operations	Count	Sum	Daily

Request metrics for tables and table buckets in CloudWatch

The `AWS/S3/Tables` namespace includes the following request metrics which are billed at the same rate as CloudWatch custom metrics. You can filter these metrics by table bucket, table, or namespace name.

> **Note**
>
> Request metrics are published only when all three dimensions (`TableBucketName`, `Namespace`, `TableName`) can be resolved. Requests that target a table or namespace that does not exist do not appear in CloudWatch request metrics. To monitor these requests, use AWS CloudTrail.

Request metrics

Metric Name	Description	Units	Statistics	Granularity
All requests count	The total number of HTTP requests made to a table bucket, regardless of type. This metric does not include `GetTableMetadataLocation` or `UpdateTableMetadataLocation` requests.	Count	Sum	1-minute
Get requests count	The number of HTTP GET requests made to retrieve objects from tables. This metric does not include `GetTableMetadataLocation` requests.	Count	Sum	1-minute
Put requests count	The number of HTTP PUT requests made to add objects to tables. This metric does not include `UpdateTableMetadataLocation` requests.	Count	Sum	1-minute
Head requests count	The number of HTTP HEAD requests made to retrieve metadata from tables.	Count	Sum	1-minute

Metric Name	Description	Units	Statistics	Granularity
Post requests count	The number of HTTP POST requests made to tables.	Count	Sum	1-minute
UpdateTableMetadataLocation requests count	The number of `UpdateTableMetadataLocation` requests made to a table bucket. This metric is not included in `Put requests count` or `All requests count`.	Count	Sum	1-minute
GetTableMetadataLocation requests count	The number of `GetTableMetadataLocation` requests made to a table bucket. This metric is not included in `Get requests count` or `All requests count`.	Count	Sum	1-minute
BytesDownloaded	The number of bytes downloaded for table requests.	Bytes	Sum	1-minute
BytesUploaded	The number of bytes uploaded for table requests.	Bytes	Sum	1-minute
4xxErrors	The count of HTTP 4xx client error status codes returned.	Count	Sum	1-minute
5xxErrors	The count of HTTP 5xx server error status codes returned.	Count	Sum	1-minute

Metric Name	Description	Units	Statistics	Granularity
FirstByte Latency	The per-request time from the complete request being received to when the response starts being returned.	Milliseconds	Sum	1-minute
TotalRequestLatency	The elapsed per-request time from the first byte received to the last byte sent.	Milliseconds	Sum	1-minute

S3 Tables dimensions in CloudWatch

The following dimensions are used to filter S3 Tables metrics.

S3 Tables dimensions

Dimension Name	Description	Example Value
`TableBucketName`	The name of the Amazon S3 table bucket	`my-table-bucket`
`Namespace`	The namespace within the table bucket that contains one or more tables	`my-department`
`TableName`	The name of a specific table within a namespace	`transactions`

Accessing CloudWatch metrics

You can monitor S3 Tables metrics using the CloudWatch Console, the AWS CLI, or the CloudWatch API. This section explains how to access your metrics using these different methods.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.

3. In the buckets list, choose the name of the bucket that contains the tables you want to see metrics for.
4. Choose the **Metrics** tab.
5. Choose the **View in CloudWatch** in any metrics pane to navigate to the CloudWatch console and see your available metrics in the `AWS/S3/Tables` namespace.

Using the AWS CLI

To list metrics for S3 Tables using the AWS CLI, use the `list-metrics` command with the `--namespace` parameter set to `AWS/S3/Tables`:

```
aws cloudwatch list-metrics --namespace AWS/S3/Tables
```

To get statistics for a specific S3 Tables metric, use the `get-metric-statistics` command. For example:

```
aws cloudwatch get-metric-statistics \
--namespace AWS/S3/Tables \
--metric-name TotalBucketStorage \
--dimensions Name=TableBucketName,Value=MyTableBucket \
--start-time 2025-03-01T00:00:00 \
--end-time 2025-03-02T00:00:00 \
--period 86400 \
--statistics Average
```

Best Practices

- When retrieving metrics, set the Period value based on the metric's granularity. For daily metrics (like storage metrics), use 86400 seconds (24 hours). For minute-level metrics (like request metrics), use 60 seconds.
- Use dimensions appropriately to filter metrics to the desired scope (table bucket, namespace, or individual table level).
- Consider using metric math to create derived metrics that better match your monitoring needs.

Related Resources

- Amazon CloudWatch Concepts

- Using Amazon CloudWatch Dashboards

Managing CloudWatch metrics

Storage metrics are enabled by default for all Amazon S3 tables and table buckets. You can enable or disable additional Request metrics through the console, AWS Command Line Interface (AWS CLI), or AWS SDKs.

Prerequisites

- Requires `s3table:PutTableBucketMetricsConfiguration` IAM permission

> **Note**
>
> S3 Tables Request metrics are billed at the same rate as CloudWatch custom metrics.

Using the AWS Management Console

To enable or disable additional metrics

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Table buckets**.
3. In the buckets list, choose the name of the table bucket that contains the tables you want to request metrics for.
4. Choose the **Metrics** tab.
5. From the Request metrics panel, choose **Edit**.
6. Select **Enabled** or **Disabled**, then **Save changes**.

Using the AWS CLI

These examples show how to enable or disable request metrics for table buckets by using the AWS CLI. To use these commands, replace the *`user input placeholders`* with your own information.

Example: To enable request metrics for a table bucket:

```
aws s3tables put-table-bucket-metrics-configuration \
```

```
--table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket
```

Example: To disable request metrics for a table bucket:

```
aws s3tables delete-table-bucket-metrics-configuration \
--table-bucket-arn arn:aws:s3tables:us-east-1:111122223333:bucket/amzn-s3-demo-table-bucket
```

Working with S3 Vectors and vector buckets

What is Amazon S3 Vectors?

Amazon S3 Vectors delivers purpose-built, cost-optimized vector storage for AI agents, inference, RAG, and semantic search. S3 Vectors is designed to provide the same elasticity, durability, and availability as Amazon S3 and delivers subsecond latency for infrequent queries and as low as 100 milliseconds for more frequent queries. You get a dedicated set of API operations to store, access, and query vector data without provisioning any infrastructure. S3 Vectors consists of several key components that work together:

- **Vector buckets** – A new bucket type that's purpose-built to store and query vectors.
- **Vector indexes** – Within a vector bucket, you can organize your vector data within vector indexes. You perform similarity queries on your vector data within vector indexes.
- **Vectors** – You store vectors in your vector index. For similarity search and AI applications, vectors are created as vector embeddings which are numerical representations that preserve semantic relationships between content (such as text, images, or audio) so similar items are positioned closer together. S3 Vectors can perform similarity searches based on semantic meaning rather than exact matching through comparing how close vectors are to each other mathematically. When adding vector data to a vector index, you can also attach metadata for future filtering queries based on a set of conditions (for example, timestamps, categories, and user preferences).

Writes to S3 Vectors are strongly consistent, which means that you can immediately access the most recently added data. As you write, update, and delete vectors over time, S3 Vectors automatically optimizes the vector data to achieve the best possible price performance for vector storage, even as the data sets scale and evolve. You can control access to your vector data with the existing access control mechanisms of Amazon S3, including bucket and IAM policies. For more information about vector index limits per bucket and vector limits per index, see Limitations and restrictions.

Use cases: Similarity searches across large datasets

Similarity searches allow you to find items that are conceptually related to each other based on their vector representations, rather than exact keyword matches. These searches identify content with similar meanings or characteristics, even when the exact words or visual elements differ.

Common use cases for similarity search with S3 Vectors include:

- **Medical imaging** - Find similarities in millions of medical images to assist with diagnosis and treatment planning
- **Copyright infringement** - Identify potentially derivative content across large media libraries
- **Image deduplication** - Detect and remove duplicate or near-duplicate images from large image collections
- **Video understanding** - Search for specific scenes or content within video assets
- **Enterprise document search** - Enable semantic search across corporate documents to find relevant information based on meaning
- **Personalization** - Deliver tailored recommendations by finding similar items

You should use S3 Vectors if you want to build cost-effective vector search and agentic AI applications with sub-second search times. With vector buckets, you only pay for what you use, and could save costs for uploading, storing, and querying vector embeddings. For more information about pricing, see Amazon S3 pricing.

Features of S3 Vectors

Purpose-built storage for vectors

S3 Vectors is the first purpose-built object storage in the cloud to store and query vectors. Vector buckets are designed to provide cost-effective, elastic, and durable storage for vector data.

Vector embeddings are transforming how customers use and retrieve their unstructured data, from detecting similarities across medical images, finding anomalies in thousands of hours of video footage, navigating through large code bases, and identifying the most relevant case law for a given legal matter. These emerging applications combine with embedding models to encode the semantic meaning of data (for example, text, images, video, code) as numerical vector embeddings.

Within a vector bucket, you organize your vector data within vector indexes, without provisioning infrastructure. As you write, update, and delete vectors over time, S3 Vectors automatically optimize the vector data to achieve the best possible price performance for vector storage, even as the data sets scale and evolve. For more information about vector index limits per bucket and vector limits per index, see Limitations and restrictions.

Perform similarity queries

With S3 Vectors, you can perform queries to find the most similar vectors to a query vector, with sub-second response times for infrequent queries and as low as 100 milliseconds for more frequent queries. S3 Vectors is ideal for workloads where queries are less frequent.

Metadata filtering

You can attach metadata (for example, year, author, genre, and location) as key-value pairs to your vectors. By default, all metadata is filterable unless you explicitly specify it as non-filterable. You can use filterable metadata to filter your query results based on specific attributes, enhancing the relevance of your queries. Vector indexes support string, number, boolean, and list types of metadata. For more information about metadata size limits per vector and filterable metadata size limits per vector, see Limitations and restrictions.

Access management and security

You can manage access for resources in vector buckets with IAM and Service Control Policies in AWS Organizations. S3 Vectors uses a different service namespace than Amazon S3: the `s3vectors` namespace. Therefore, you can design policies specifically for the S3 Vectors service and its resources. You can design policies to grant access to individual vector indexes, all vector indexes within a vector bucket, or all vector buckets in an account. All Amazon S3 Block Public Access settings are always enabled for vector buckets and cannot be disabled.

Integration with AWS services

S3 Vectors integrates with other AWS services to enhance your vector processing capabilities:

- **Amazon OpenSearch Service** - Optimize vector storage costs while continuing to use OpenSearch API operations. This is ideal for workloads that need advanced search functionality such as hybrid search, aggregations, advanced filtering, and faceted search. You can also export a snapshot of an S3 vector index to Amazon OpenSearch Serverless for high QPS and low latency vector search.
- **Amazon Bedrock Knowledge Bases** - Select a vector index in S3 Vectors as your vector store to save on storage costs for retrieval augmented generation (RAG) applications.
- **Amazon Bedrock in SageMaker Unified Studio** - Develop and test knowledge bases using S3 Vectors as your vector store.

Tutorial: Getting started with S3 Vectors

In this tutorial, you create an S3 vector bucket and a vector index in an AWS Region in the Amazon S3 console. Next, you use an Amazon Bedrock embedding model to generate vector embeddings of your data and store them in your vector index to perform semantic searches.

To get started, sign up for an AWS account if you don't already have one. For more information about setting up with an account, see Getting started with Amazon S3.

Topics

- Step 1: Create a vector bucket with the console
- Step 2: Create a vector index in a vector bucket with the console
- Step 3: Insert vectors into a vector index with the SDK for Python (Boto3)
- Step 4. Query vectors in a vector index with the SDK for Python (Boto3)
- (Optional) Automate vector embedding creation with the S3 Vectors Embed CLI
- (Optional) Integrate S3 Vectors with Amazon Bedrock Knowledge Bases
- (Optional) Integrate S3 Vectors with Amazon OpenSearch

Step 1: Create a vector bucket with the console

S3 vector buckets are a type of Amazon S3 bucket that's purpose-built to store and query vectors. You use dedicated API operations to write and query vector data. You can control access to your vector data with access control mechanisms such as IAM identity-based policies and resource-based policies.

In this step, you use the Amazon S3 console to create your first vector bucket. For other ways to create a vector bucket, see Creating a vector bucket.

To create a vector bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Vector buckets**.
3. On the **Vector buckets** page, choose **Create vector bucket**.
4. On the **Create vector bucket** page, for the **Vector bucket name** field, enter a name. In this tutorial, we use `media-embeddings` as the vector bucket name. Vector bucket names must be

3 to 63 characters and unique within an AWS Region. Valid characters are lowercase letters (a-z), numbers (0-9), and hyphens (-).

> **Note**
>
> The vector bucket name can't be changed after the bucket is created.

5. Under **Encryption**, choose **Specify encryption type**. You have the option to specify encryption type as **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** or the default **Server-side encryption with Amazon S3 managed keys (SSE-S3)**. In this tutorial, we choose **Server-side encryption with Amazon S3 managed keys (SSE-S3)**. For more information about setting encryption configuration for vector buckets, see Data protection and encryption in S3 Vectors.

> **Note**
>
> The encryption type can't be changed after the vector bucket is created.

6. Under **Tags (Optional)**, you can add tags as key-value pairs to help organize and specify permissions for your vector buckets. Enter a **Key** and a **Value**. To add another tag, choose **Add Tag**. You can enter up to 50 tags for a vector bucket. For more information, see Using tags with S3 vector buckets.
7. Choose **Create vector bucket**.

Step 2: Create a vector index in a vector bucket with the console

> **Note**
>
> Choose your vector index configuration parameters carefully. After you create a vector index, you can't update the vector index name, dimension, distance metric, or non-filterable metadata keys. To change any of these values, you must create a new vector index.

In this step, you create a vector index in your vector bucket. The vector index is used to store and query your vector embeddings. For more information about vector indexes, see Vector indexes.

To create a vector index

1. Navigate to the newly created vector bucket in the **Vector buckets** list on the Amazon S3 console.
2. Choose **Create vector index**.
3. On the **Create vector index** page, enter a name in the **Vector index name** field. In this tutorial, we use *`movies`* as the vector index name.

   Vector index names must be 3 to 63 characters and unique within this vector bucket. Valid characters are lowercase letters (a-z), numbers (0-9), hyphens (-), and dots (.).

   The vector index name can't be changed after index creation.
4. Under **Dimension**, specify a numeric value between 1 and 4096 that determines how many numbers will be in each vector that's generated by your vector embedding model. Embedding models are specialized machine learning (ML) models that convert data (such as text or images) into numerical vectors. Embedding models typically produce outputs between 500-2000 dimensions, with each dimension being a floating-point number. In this tutorial, we'll use the Titan Text v2 embedding model from Amazon Bedrock. This embedding model uses 1024-dimensional vectors as the default, so we'll set dimension for the vector index to `1024`.

   The dimension value can't be changed after index creation. Carefully specify the dimension value based on your embedding model's recommendations.
5. Under **Distance metric**, choose either **Cosine** (which measures angular similarity) or **Euclidean** (which measures straight-line distance) as the distance metric to define how similarity between vectors is calculated during queries. In this tutorial, we choose **Cosine**.

   The distance metric can't be changed after index creation. Carefully choose the distance metric based on your embedding model's recommendations.
6. Under **Additional settings (optional)**, you can specify non-filterable metadata keys to store supplementary information alongside your vector embeddings.

   > **Note**
   >
   > You can attach filterable metadata as key-value pairs to each vector when you insert vector data after index creation. By default, all metadata keys that are attached to vectors are filterable and can be used as filters in a similarity query. Only metadata

keys that are specified as non-filterable during vector index creation are excluded from filtering.

Each non-filterable metadata key must be between 1 to 63 characters and unique within the vector index. These keys are useful for storing reference information that you don't need to filter on during similarity queries. For example, when working with text embeddings, you might want to preserve the original text chunks for reference purposes. In this tutorial, we add a non-filterable metadata key that's named `source_text` to store the text data that we want to create vector embeddings for.

You can't update non-filterable metadata keys after index creation. Plan your metadata structure carefully.

7. Under **Encryption**, choose **Specify encryption type**. You have the option to use the bucket level encryption settings or override the encryption settings for the vector index. If you override the bucket-level settings, you have the option to specify encryption type for the vector index as **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** or the **Server-side encryption with Amazon S3 managed keys (SSE-S3)**. In this tutorial, we choose to **Use bucket settings for encryption**. For more information about setting encryption configuration for vector buckets and indexes, see Data protection and encryption in S3 Vectors.
8. Under **Tags (Optional)**, you can add tags as key-value pairs to help track and organize vector index costs using AWS Billing and Cost Management. Enter a **Key** and a **Value**. To add another tag, choose **Add Tag**. You can enter up to 50 tags for a vector index. For more information, see Using tags with S3 vector buckets.
9. Choose **Create vector index**.

Verify that your new vector index appears inside the bucket.

Step 3: Insert vectors into a vector index with the SDK for Python (Boto3)

To insert, list, and query vectors, use the AWS SDKs, AWS CLI, or Amazon S3 REST API.

This step is to insert vector embeddings to your vector index with the PutVectors API operation.

You must first generate vector embeddings by using the embedding model of your choice. If you use Amazon Bedrock, generate embeddings by using the InvokeModel API operation to specify your preferred embedding model. Alternatively, to generate and insert vector embeddings with a single command, see (Optional) Automate vector embedding creation with the S3 Vectors Embed CLI.

The following example code generates 1024-dimensional vector embeddings with the Titan Text Embeddings V2 model from Amazon Bedrock by using the AWS SDK for Python (Boto3), and stores them in your vector index by using the `PutVectors` API. Along with each vector, we attach key-value pairs as filterable metadata. Additionally, we use a non-filterable metadata key that's named *`source_text`* to preserve the original text from which each vector is derived. To maximize request throughput and optimize for speed and efficiency, we recommend that you insert and delete vectors in batches. For more information, see S3 Vectors best practices.

```
# Populate a vector index with embeddings from Amazon Titan Text Embeddings V2.
import boto3
import json

# Create Bedrock Runtime and S3 Vectors clients in the AWS Region of your choice.
bedrock = boto3.client("bedrock-runtime", region_name="us-west-2")
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

# Texts to convert to embeddings.
texts = [
    "Star Wars: A farm boy joins rebels to fight an evil empire in space",
    "Jurassic Park: Scientists create dinosaurs in a theme park that goes wrong",
    "Finding Nemo: A father fish searches the ocean to find his lost son"
]

# Generate vector embeddings.
embeddings = []
for text in texts:
    response = bedrock.invoke_model(
        modelId="amazon.titan-embed-text-v2:0",
        body=json.dumps({"inputText": text})
    )

    # Extract embedding from response.
    response_body = json.loads(response["body"].read())
    embeddings.append(response_body["embedding"])

# Write embeddings into vector index with metadata.
```

```
s3vectors.put_vectors(
    vectorBucketName="media-embeddings",
    indexName="movies",
    vectors=[
        {
            "key": "Star Wars",
            "data": {"float32": embeddings[0]},
            "metadata": {"source_text": texts[0], "genre":"scifi"}
        },
        {
            "key": "Jurassic Park",
            "data": {"float32": embeddings[1]},
            "metadata": {"source_text": texts[1], "genre":"scifi"}
        },
        {
            "key": "Finding Nemo",
            "data": {"float32": embeddings[2]},
            "metadata": {"source_text": texts[2], "genre":"family"}
        }
    ]
)
```

Step 4. Query vectors in a vector index with the SDK for Python (Boto3)

After you store your vectors embeddings in the vector index, you can perform a similarity search by using the QueryVectors API operation.

You must first generate your query vector embeddings by using the same embedding model that's used during insertion in Step 3: Insert vectors into a vector index with the SDK for Python (Boto3). In this tutorial, use the Titan Text Embeddings V2 model from Amazon Bedrock through the SDK for Python (Boto3).

You can perform a similarity search to return the closest matching vectors. In a similarity search, you can optionally filter by using metadata keys to narrow results based on specific criteria while maintaining semantic relevance.

```
# Query a vector index with an embedding from Amazon Titan Text Embeddings V2.
import boto3
import json

# Create Bedrock Runtime and S3 Vectors clients in the AWS Region of your choice.
bedrock = boto3.client("bedrock-runtime", region_name="us-west-2")
```

```
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

# Query text to convert to an embedding.
input_text = "adventures in space"

# Generate the vector embedding.
response = bedrock.invoke_model(
    modelId="amazon.titan-embed-text-v2:0",
    body=json.dumps({"inputText": input_text})
)

# Extract embedding from response.
model_response = json.loads(response["body"].read())
embedding = model_response["embedding"]

# Query vector index.
response = s3vectors.query_vectors(
    vectorBucketName="media-embeddings",
    indexName="movies",
    queryVector={"float32": embedding},
    topK=3,
    returnDistance=True,
    returnMetadata=True
)
print(json.dumps(response["vectors"], indent=2))

# Query vector index with a metadata filter.
response = s3vectors.query_vectors(
    vectorBucketName="media-embeddings",
    indexName="movies",
    queryVector={"float32": embedding},
    topK=3,
    filter={"genre": "scifi"},
    returnDistance=True,
    returnMetadata=True
)
print(json.dumps(response["vectors"], indent=2))
```

(Optional) Automate vector embedding creation with the S3 Vectors Embed CLI

Amazon S3 Vectors Embed CLI is a standalone command line tool that simplifies the process of working with vector embeddings in S3 Vectors. With single commands, you can create vector embeddings for your data by using Amazon Bedrock, and store and query them in your S3 vector index. The tool supports the following commands:

- `s3vectors-embed put`: Generate vector embeddings of text, file content, or S3 objects, and store them as vectors in an S3 vector index.
- `s3vectors-embed query`: Embed a query input and search for similar vectors in an S3 vector index.

For more information about the commands and usage, see the Amazon S3 Vectors Embed CLI GitHub repository.

(Optional) Integrate S3 Vectors with Amazon Bedrock Knowledge Bases

Amazon Bedrock Knowledge Bases integrates with S3 Vectors for cost effective and long-term storage of vector datasets.

Prerequisites

Follow the prerequisites to ensure you have necessary permissions for creating an Amazon Bedrock knowledge base.

- Appropriate IAM permissions for both S3 Vectors and Amazon Bedrock services. For more information about IAM permissions for S3 Vectors, see Identity and Access management in S3 Vectors.
- Your source documents prepared for ingestion into the knowledge base.
- An understanding of your embedding model requirements.

Create an Amazon Bedrock knowledge base with S3 Vectors in the console

Amazon Bedrock Knowledge Bases provides a fully managed end-to-end RAG workflow. When you create a knowledge base with S3 Vectors, Amazon Bedrock automatically fetches data from your S3 data source, converts content into text blocks, generates embeddings, and stores them in your

vector index. You can then query the knowledge base and generate responses based on chunks retrieved from your source data.

To create an Amazon Bedrock knowledge base with S3 Vectors in the console

1. Sign in to the AWS Management Console using an IAM role with Amazon Bedrock permissions, and open the Amazon Bedrock console at https://console.aws.amazon.com/bedrock/.
2. In the left navigation pane, choose **Knowledge Bases**.
3. In the **Knowledge Bases** section, choose **Create**. In the dropdown list, choose **Knowledge Base with vector store**.
4. Keep the **Knowledge Base name**, **Knowledge Base description**, and **IAM permissions** at their default values in this tutorial.
5. For **Choose a data source type**, choose the data source to connect your knowledge base to. In this tutorial, choose **Amazon S3**.
6. (Optionally) Provide any **Tags** or **Application logs**.
7. Choose **Next**.
8. On the **Configure data source** page, keep the **Data source name**, **Data source location**, **Parsing strategy**, and **Chunking strategy** at the default settings in this tutorial.
9. In the **S3 URI**, enter the S3 general purpose bucket that contains your source files, or choose **Browse S3** and choose the S3 general purpose bucket.
10. For encrypted S3 data, select **Add customer-managed KMS key for S3 data** and specify your customer-managed KMS key.
11. Choose **Next** to proceed to the next step that specifies the **Embeddings model** and **Vector store** options.
12. In the **Embeddings model** section, choose **Select model** and choose an embedding model compatible for use with S3 Vectors. You must use floating-point embeddings. Binary type isn't supported.
13. (Optional) Expand the **Additional configurations** section to see the following configuration options (not all models support all configurations):

    - **Embeddings type** – Whether to convert the data to floating-point (float32) vector embeddings (more precise, but more costly) or binary vector embeddings (less precise, but less costly). To integrate with S3 Vectors, you must choose **Floating-point vector embeddings**.
    - **Vector dimensions** – Choose the embedding models recommended dimension size.

14. In the **Vector store** section, choose the recommended method **Quick create a new vector store** for automatic setup of a new vector bucket, or choose **Use an existing vector store** to use an existing vector bucket if available. For information about using an existing vector store flow, see Prerequisites for using a vector store you created for a knowledge base in the *Amazon Bedrock User Guide*.
15. For **Additional configurations** – By default, your vector bucket uses server-side encryption with Amazon S3 managed keys (SSE-S3). You can choose to use your own KMS key for server-side encryption with AWS Key Management Service keys (SSE-KMS) for enhanced control over encryption settings.
16. Choose **Next** to review the Knowledge Base details and then choose **Create Knowledge Base**.

> **Note**
>
> The time that it takes to create a knowledge base depends on your specific configurations. When the creation of a knowledge base completes, the status of the knowledge base changes to state it is either ready or available. Once your knowledge base is ready and available, sync your data source for the first time or whenever you want to keep your content up to date. To sync your data source, choose your knowledge base in the console and choose **Sync** within the data source overview section.

(Optional) Integrate S3 Vectors with Amazon OpenSearch

Amazon OpenSearch Service is a fully managed service that simplifies the deployment, scaling, and operation of OpenSearch in the AWS Cloud. There are two integrations between S3 Vectors and OpenSearch. One is to export vector data from S3 Vectors to OpenSearch Serverless for high-performance search capabilities. The other uses S3 Vectors as a cost-effective storage engine within OpenSearch while maintaining access to OpenSearch functionality.

For more information, see Using S3 Vectors with OpenSearch Service.

Vector buckets

Vector buckets are a type of Amazon S3 bucket designed specifically for storing and querying vector data. Vector buckets use dedicated APIs to manage vector data efficiently and reduce costs of upload, storing, and querying vector embeddings. Vector buckets provide the foundation for

organizing your vector data into indexes, enabling you to perform similarity searches across large datasets while benefiting from the availability, durability, scalability, and cost-effectiveness of Amazon S3.

Vector buckets are optimized for long-term vector storage with sub-second search times. You can perform similarity queries on your vector data and optionally attach metadata to filter queries based on specific conditions such as dates, categories, or user preferences.

Each vector bucket has a unique Amazon Resource Name (ARN) and resource policy attached to it. The ARNs of vector buckets follow the following format:

```
arn:aws:s3vectors:Region:OwnerAccountID:bucket/bucket-name
```

Within a vector bucket, you create vector indexes to store and query your data. Each vector bucket exists within a specific AWS Region and you can create multiple vector indexes inside a vector bucket. Vector buckets support security and access control mechanisms, including IAM identity-based policies and bucket policies. You can use bucket policies to grant or restrict access to specific indexes within your vector bucket.

Key characteristics of vector buckets:

- Purpose-built for vector storage and similarity search operations.
- Strongly consistent writes ensure that the vector data is immediately accessible.
- Automatic optimization of vector data for best price-performance as datasets scale.

For more information about vector index limits per bucket and other limitations, see Limitations and restrictions.

Topics

- Vector bucket naming rules
- Creating a vector bucket
- Listing vector buckets
- Viewing vector bucket attributes
- Deleting an empty vector bucket
- Managing vector bucket policies
- Using tags with S3 vector buckets

Vector bucket naming rules

Vector bucket names must follow specific naming conventions to ensure uniqueness within an AWS Region. Amazon S3 enforces the following bucket naming requirements, and you can't create a vector bucket if these rules aren't followed. Additionally, there are best practices that, while not enforced, help prevent conflicts when working with vector buckets programmatically or through the console.

Vector bucket naming requirements

When creating vector buckets, you must follow these requirements:

- Vector bucket names must be unique in the same AWS account for each AWS Region.
- Vector bucket names must be between 3 and 63 characters long.
- Vector bucket names can consist only of lowercase letters (a-z), numbers (0-9), and hyphens (-).
- Vector bucket names must begin and end with a letter or number.

Best practices for naming

We recommend following these best practices when naming your vector buckets:

- Use descriptive names that reflect the purpose of your vector data (for example, product-recommendations, document-embeddings).
- Avoid using sensitive information in bucket names as they may appear in logs and URLs.
- Keep names concise but meaningful for easier management and identification.

These naming conventions ensure that your vector buckets can be reliably accessed through the AWS Management Console, Amazon S3 REST API, the AWS CLI, and AWS SDKs.

Creating a vector bucket

You can create a vector bucket using the S3 console or the AWS CLI. All data stored in vector buckets are always encrypted at rest. By default, vector buckets use SSE-S3 to encrypt vector data. You can choose to configure buckets to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) instead. The bucket encryption settings can’t be changed after a vector bucket is created, so it's important to choose the appropriate encryption method based on

your security requirements and compliance needs. For more information about security in vector buckets, see Data protection and encryption in S3 Vectors.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Vector buckets**.
3. Choose **Create vector bucket**.
4. For **Vector bucket name**, enter a name for your bucket.

   The bucket name must follow the naming rules:

   - Bucket name must be between 3 and 63 characters long.
   - Bucket name can only include lowercase letters, numbers, and hyphens.
   - Bucket name must be unique within your AWS account for an AWS Region.

   For more information about vector bucket naming rules, see Vector bucket naming rules.

   > **Important**
   >
   > You can't change the vector bucket name after you create the bucket.

5. For **Encryption**, choose one of the following options:

   - **Don't specify encryption type** – Amazon S3 automatically applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for new vectors. Choose this option for the simplest setup with no additional configuration.
   - **Specify encryption type** – Choose a specific encryption method:
     - **Server-side encryption with Amazon S3 managed keys (SSE-S3)** – Explicitly choose to use SSE-S3. Amazon S3 encrypts your vector data as it writes it to storage and decrypts it when you access it. AWS manages all encryption keys automatically.
     - **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** – Uses customer managed keys (CMKs) in AWS KMS, giving you more control over your encryption keys, key rotation, and access policies.

       If you select SSE-KMS, you have additional options:

- **Choose from your AWS KMS keys** – Select an existing customer managed key from your account.
- **Enter AWS KMS key ARN** – Specify the full ARN of a KMS key (required format).
- **Create a KMS key** – Opens the AWS KMS console to create a new customer managed key.

KMS key requirements:

- The KMS key must be in the same Region as the vector bucket.
- You must specify the full KMS key ARN (key IDs and aliases aren't supported).
- You must grant the S3 Vectors service principal (`indexing.s3vectors.amazonaws.com`) the `kms:Decrypt` permission to use the key. For more information about an example AWS KMS key policy, see Data protection and encryption in S3 Vectors.

For detailed information about encryption options and KMS key setup, see Using SSE-KMS encryption.

> **Important**
>
> Encryption settings can't be changed after the vector bucket is created. Choose carefully based on your long-term security and compliance requirements.

6. Under **Tags (Optional)**, you can add tags as key-value pairs to help track and organize vector index costs using AWS Billing and Cost Management. Enter a **Key** and a **Value**. To add another tag, choose **Add Tag**. You can enter up to 50 tags for a vector index. For more information, see Using tags with S3 vector buckets.
7. Choose **Create vector bucket**.

After creation, you'll see a confirmation message. The new vector bucket appears in your vector buckets list and is ready for creating vector indexes within the bucket.

Using the AWS CLI

You can create a vector bucket with SSE-S3 encryption using the following command. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3vectors create-vector-bucket \
   --vector-bucket-name "amzn-s3-demo-vector-bucket"
```

To create a vector bucket with SSE-KMS encryption using a customer managed KMS key:

```
aws s3vectors create-vector-bucket \
   --vector-bucket-name "amzn-s3-demo-vector-bucket" \
   --encryption-configuration '{"sseType": "aws:kms", "kmsKeyArn": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"}'
```

Using the AWS SDKs

SDK for Python (Boto3)

```
import boto3

# Create a S3 Vectors client in the AWS Region of your choice.
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

#Create a vector bucket
s3vectors.create_vector_bucket(vectorBucketName="media-embeddings")
```

Listing vector buckets

You can view all your vector buckets using the Amazon S3 console, AWS CLI, or AWS SDKs. The listing operations support prefix-based filtering to help you find specific buckets when you have many vector buckets in your account. For more information about `ListVectorBuckets`, prefix limits, and response limits, see ListVectorBuckets in the *Amazon S3 API Reference*.

Prefix search capability

Prefix search allows you to list buckets that start with a specific prefix, making it easier to organize and find related vector buckets. This is particularly useful when you use naming conventions that group related buckets together:

- **Environment-based**: `production-vectors-`, `staging-vectors-`, `dev-vectors-`
- **Use case-based**: `ml-model-vectors-`, `document-search-`, `image-similarity-`
- **Team-based**: `data-science-vectors-`, `ml-platform-vectors-`

Using the S3 console

To list vector buckets

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Vector buckets**.

   The console displays a list of all your vector buckets with the following information:

   - **Name** – The unique name of the vector bucket
   - **Creation date** – When the bucket was created
   - **Amazon Resource Name (ARN)** – The full ARN for programmatic access

To filter the list:

- To find a bucket based on the start of the bucket name, enter a vector bucket name or prefix in the search box above the bucket list.
- Use prefixes to find groups of related buckets (for example, type "prod-" to find all production buckets)

  The list updates in real-time as you type

Using the AWS CLI

```
aws s3vectors list-vector-buckets
```

Using the AWS SDKs

SDK for Python (Boto3)

```
import boto3

# Create a S3 Vectors client in the AWS Region of your choice.
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

#List vector buckets
response = s3vectors.list_vector_buckets()
buckets = response["vectorBuckets"]
```

```
print(buckets)
```

Viewing vector bucket attributes

You can view detailed information about a vector bucket, including its properties, encryption settings and creation details using the Amazon S3 REST API, AWS SDKs, S3 Console, or the AWS Command Line Interface (AWS CLI). For more information about `GetVectorBucket`, see GetVectorBucket in the *Amazon S3 API Reference*.

Using the S3 console

1. Sign in to the console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Vector buckets**.
3. The console displays a list of all your vector buckets. Find a bucket based on the start of the bucket name, enter a vector bucket name or prefix in the search box above the bucket list. Once you locate your vector bucket, you can view detailed information about it, including its encryption settings, tags, default index mode for new vector indexes, and creation details in the **Properties** tab.

Using the AWS CLI

```
aws s3vectors get-vector-bucket --vector-bucket-name "amzn-s3-demo-vector-bucket"
```

Deleting an empty vector bucket

You can delete a vector bucket when you no longer need it. To delete a vector bucket, you must first delete all the vector indexes within the bucket. When you delete a vector index, all vector data within it is deleted. Use the Amazon S3 REST API, AWS SDKs, S3 Console, or the AWS Command Line Interface (AWS CLI) to delete a vector bucket.

Before you can delete a vector bucket, you must:

- Delete all vector indexes in the bucket.
- Ensure no operations are in progress on the bucket or its indexes.

Important

- Bucket deletion is permanent and can't be undone.
- All data and configuration associated with the bucket is permanently lost.
- The bucket name becomes available for reuse after deletion.
- Any applications or scripts referencing the bucket will receive errors after deletion.

Using the S3 console

1. Sign in to the console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Vector buckets**.
3. The console displays a list of all your vector buckets. Find a bucket based on the start of the bucket name, enter a vector bucket name or prefix in the search box above the bucket list. Once you locate and choose your vector bucket, select the **Delete** option.
4. To confirm this deletion, type **`delete`** and then select **Delete vector bucket**.

Using the AWS CLI

First, check that no vector indexes exist in the bucket. For more information about how to verify the bucket is empty, see Listing vector indexes.

If indexes exist, you must delete all vectors from each index and then delete the indexes. For more information about how to verify the bucket is empty, see Listing vectors, Deleting vectors from a vector index, and Deleting a vector index.

To delete the empty vector bucket, use the following example command and replace the *`user input placeholders`* with your own information.

```
aws s3vectors delete-vector-bucket \
  --vector-bucket-name "amzn-s3-demo-vector-bucket"
```

Using the AWS SDKs

SDK for Python (Boto3)

```
import boto3

# Create a S3 Vectors client in the AWS Region of your choice.
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

#Delete a vector bucket
response = s3vectors.delete_vector_bucket(vectorBucketName="media-embeddings")
```

Managing vector bucket policies

Vector bucket policies are resource-based policies that you attach directly to vector buckets to control access to the bucket and its contents. You can add, view, edit, delete vector bucket policies by using the Amazon S3 REST API, AWS SDKs, S3 Console, or the AWS Command Line Interface (AWS CLI). Bucket policies for vector buckets can grant permissions to principals from other AWS accounts, making them useful for cross-account access scenarios.

Policy management operations

- PutVectorBucketPolicy – Add or update a bucket policy.
- GetVectorBucketPolicy – Retrieve the current bucket policy.
- DeleteVectorBucketPolicy – Remove the bucket policy.

Adding a vector bucket policy

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Amazon S3**.
3. Choose **Vector buckets** and select the vector bucket name that you want to add a policy to.
4. Choose the **Permissions** tab.
5. Under **Vector bucket policy**, choose **Edit**.
6. In the policy editor, enter your policy JSON.

7. (Optional) Choose **Policy examples** to see sample policies that you can adapt to your needs.
8. After entering your policy, choose **Save changes**.

Using the AWS CLI

To add or update a bucket policy, use the following example command and replace the *user input placeholders* with your own information.

```
aws s3vectors put-vector-bucket-policy \
  --vector-bucket-name "amzn-s3-demo-vector-bucket" \
  --policy '{"Version": "2012-10-17",       "Statement":[{"Effect":"Allow","Principal":
{"AWS":"arn:aws:iam::111122223333:root"},"Action":"s3vectors:*","Resource":"arn:aws:s3vectors:a
region:111122223333:bucket/amzn-s3-demo-vector-bucket"}]}'
```

Viewing a vector bucket policy

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Amazon S3**.
3. Choose **Vector buckets** and select the vector bucket name that you want to view the policy for.
4. Choose the **Permissions** tab.

Using the AWS CLI

To retrieve a bucket policy, use the following example command and replace the *user input placeholders* with your own information.

```
aws s3vectors get-vector-bucket-policy \
  --vector-bucket-name "amzn-s3-demo-vector-bucket"
```

Deleting a vector bucket policy

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Amazon S3**.

3. Choose **Vector buckets** and select the vector bucket name that you want to delete the policy for.
4. Choose the **Permissions** tab.
5. Under the **Vector bucket policy**, choose **Delete**.

Using the AWS CLI

To delete a bucket policy, use the following example command and replace the *`user input placeholders`* with your own information.

```
aws s3vectors delete-vector-bucket-policy \
  --vector-bucket-name "amzn-s3-demo-vector-bucket"
```

For detailed information about creating and managing bucket policies, including policy examples and best practices, see S3 Vectors resource-based policy examples.

Using tags with S3 vector buckets

An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 vector buckets. You can tag S3 vector buckets when you create them or manage tags on existing vector buckets. For general information about tags, see Tagging for cost allocation or attribute-based access control (ABAC).

> **Note**
>
> There is no additional charge for using tags on vector buckets beyond the standard S3 API request rates. For more information, see Amazon S3 pricing.

Common ways to use tags with vector buckets

Use tags on your S3 vector buckets for:

- **Attribute-based access control (ABAC)** – Scale access permissions and grant access to S3 vector buckets based on their tags. For more information, see Using tags for attribute-based access control (ABAC).

ABAC for S3 vector buckets

Amazon S3 vector buckets support attribute-based access control (ABAC) using tags. Use tag-based condition keys in your AWS organizations, IAM, and S3 vector bucket policies. For enterprises, ABAC inAmazon S3 supports authorization across multiple AWS accounts.

In your IAM policies, you can control access to S3 vector buckets based on the vector bucket's tags by using the following global condition keys:

`aws:ResourceTag/key-name`

Use this key to compare the tag key-value pair that you specify in the policy with the key-value pair attached to the resource. For example, you could require that access to a resource is allowed only if the resource has the attached tag key `Dept` with the value `Marketing`. For more information, see Controlling access to AWS resources.

`aws:RequestTag/key-name`

Use this key to compare the tag key-value pair that was passed in the request with the tag pair that you specify in the policy. For example, you could check whether the request includes the tag key `Dept` and that it has the value `Accounting`. For more information, see Controlling access during AWS requests. You can use this condition key to restrict which tag key-value pairs can be passed during the `TagResource` and `CreateVectorBucket` API operations.

`aws:TagKeys`

Use this key to compare the tag keys in a request with the keys that you specify in the policy. We recommend that when you use policies to control access using tags, use the `aws:TagKeys` condition key to define what tag keys are allowed. For example policies and more information, see Controlling access based on tag keys. You can create an S3 vector bucket with tags. To allow tagging during the `CreateVectorBucket` API operation, you must create a policy that includes both the `s3vectors:TagResource` and `s3vectors:CreateVectorBucket` actions. You can then use the `aws:TagKeys` condition key to enforce using specific tags in the `CreateVectorBucket` request.

`s3vectors:VectorBucketTag/tag-key`

Use this condition key to grant permissions to specific data in vector buckets using tags. This condition key acts on the tags assigned to the vector bucket for all S3 Vectors actions. Even when you create a index with tags, this condition key acts on the tags applied to the vector bucket that contains that index. For example, you could require that access to a bucket is allowed only if the bucket has the attached tag key `Dept` with the value `Marketing`. When

accessing indexes, this condition references tags associated with the vector bucket containing that index, while the `aws:ResourceTag/tag-key` will reference the tags of the index itself.

Example ABAC policies for vector buckets

See the following example ABAC policies for Amazon S3 vector buckets.

1.1 - IAM policy to create or modify vector buckets with specific tags

In this IAM policy, users or roles with this policy can only create S3 vector buckets if they tag the vector bucket with the tag key `project` and tag value `Trinity` in the vector bucket creation request. They can also add or modify tags on existing S3 vector buckets as long as the `TagResource` request includes the tag key-value pair `project:Trinity`. This policy does not grant read, write, or delete permissions on the vector buckets or its objects.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateVectorBucketWithTags",
      "Effect": "Allow",
      "Action": [
        "s3vectors:CreateVectorBucket",
        "s3vectors:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/project": [
            "Trinity"
          ]
        }
      }
    }
  ]
}
```

1.2 - Vector bucket policy to restrict operations on the vector bucket using tags

In this vector bucket policy, IAM principals (users and roles) can perform operations using the `PutVectorBucketPolicy` action on the vector bucket only if the value of the vector bucket's `project` tag matches the value of the principal's `project` tag.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowObjectOperations",
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "s3vectors:PutVectorBucketPolicy",
      "Resource": "arn:aws:s3vectors:us-west-2:111122223333:bucket/amzn-s3-demo-vector-
bucket",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        }
      }
    }
  ]
}
```

1.3 - IAM policy to modify tags on existing resources maintaining tagging governance

In this IAM policy, IAM principals (users or roles) can modify tags on a vector bucket only if the value of the vector bucket's `project` tag matches the value of the principal's `project` tag. Only the four tags `project`, `environment`, `owner`, and `cost-center` specified in the `aws:TagKeys` condition keys are permitted for these vector buckets. This helps enforce tag governance, prevents unauthorized tag modifications, and keeps the tagging schema consistent across your vector buckets.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceTaggingRulesOnModification",
      "Effect": "Allow",
      "Action": [
        "s3vectors:TagResource"
      ],
      "Resource": "arn:aws:s3vectors:us-west-2:111122223333:bucket/*",
      "Condition": {
        "StringEquals": {
```

```
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "project",
            "environment",
            "owner",
            "cost-center"
          ]
        }
      }
    }
  ]
}
```

1.4 - Using the `s3vectors:VectorBucketTag` condition key

In this IAM policy, the condition statement allows access to the vector bucket's and vector index's operations only if the vector bucket has the tag key `Environment` and tag value `Production`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToTaggedBucket",
      "Effect": "Allow",
      "Action": "*",
      "Resource": "arn:aws:s3vectors:us-west-2:111122223333:bucket/*",
      "Condition": {
        "StringEquals": {
          "s3vectors:VectorBucketTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Managing tags for vector buckets

You can add or manage tags for S3 vector buckets using the Amazon S3 Console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or using the S3 APIs: TagResource, UntagResource, and ListTagsForResource. For more information, see:

Creating vector buckets with tags

You can tag Amazon S3 vector buckets when you create them. There is no additional charge for using tags on vector buckets beyond the standard S3 API request rates. For more information, see Amazon S3 pricing. For more information about tagging vector buckets, see Using tags with S3 vector buckets.

Permissions

To create a vector bucket with tags, you must have the following permissions:

- `s3vectors:CreateVectorBucket`
- `s3vectors:TagResource`

Troubleshooting errors

If you encounter an error when attempting to create a vector bucket with tags, you can do the following:

- Verify that you have the required Permissions to create the vector bucket and add a tag to it.
- Check your IAM user policy for any attribute-based access control (ABAC) conditions. You may be required to label your vector buckets only with specific tag keys and values. For more information, see Using tags for attribute-based access control (ABAC).

Steps

You can create a vector bucket with tags applied by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To create a vector bucket with tags using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **vector buckets**.
3. Choose **create vector bucket** to create a new vector bucket.
4. Create a vector bucket as you normally would; see Creating a vector bucket.

5. On the **Create vector bucket** page, **Tags** is an option when creating a new vector bucket.
6. Enter a name for the vector bucket.
7. Choose **Add new Tag** to open the Tags editor and enter a tag key-value pair. The tag key is required, but the value is optional.
8. To add another tag, select **Add new Tag** again. You can enter up to 50 tag key-value pairs.
9. After you complete specifying the options for your new vector bucket, choose **Create vector bucket**.

Using the REST API

For information about the Amazon S3 REST API support for creating a vector bucket with tags, see the following section in the *Amazon S3 Vectors API Reference*:

CreateVectorBucket

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to create a vector bucket with tags by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

When you create a vector bucket you must provide configuration details and use the following naming convention: `example-vector-bucket`

```
aws s3vector create-vector-bucket --vector-bucket-name acc-bucket \
  --tags Department=Accounting,Stage=Prod
```

Adding a tag to a vector bucket

You can add tags to Amazon S3 vector buckets and modify these tags. There is no additional charge for using tags on vector buckets beyond the standard S3 API request rates. For more information, see Amazon S3 pricing. For more information about tagging vector buckets, see Using tags with S3 vector buckets.

Permissions

To add a tag to a vector bucket, you must have the following permission:

- `s3vectors:TagResource`

Troubleshooting errors

If you encounter an error when attempting to add a tag to a vector bucket, you can do the following:

- Verify that you have the required Permissions to add a tag to a vector bucket.
- If you attempted to add a tag key that starts with the AWS reserved prefix `aws:`, change the tag key and try again.

Steps

You can add tags to vector buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWSSDKs.

Using the S3 console

To add tags to a vector bucket using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **vector buckets**.
3. Choose the vector bucket name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and choose **Add new Tag**.
6. This opens the **Add Tags** page. You can enter up to 50 tag key value pairs.
7. If you add a new tag with the same key name as an existing tag, the value of the new tag overrides the value of the existing tag.
8. You can also edit the values of existing tags on this page.
9. After you have added the tag(s), choose **Save changes**.

Using the REST API

For information about the Amazon S3 REST API support for adding tags to a vector bucket, see the following section in the *Amazon S3 Vectors API Reference*:

TagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to add tags to a vector bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

```
aws s3vectors tag-resource \
--resource-arn arn:aws:s3vectors:us-east-1:012345678900:bucket/acc-bucket \
--tags Stage=Prod,CostCenter=Marketing
```

Viewing vector bucket tags

You can view or list tags applied to Amazon S3 vector buckets. For more information about tagging vector buckets, see Using tags with S3 vector buckets.

Permissions

To view tags applied to a vector bucket, you must have the following permission:

- `s3vectors:ListTagsForResource`

Troubleshooting errors

If you encounter an error when attempting to list or view the tags of a vector bucket, you can do the following:

- Verify that you have the required Permissions to view or list the tags of the vector bucket.

Steps

You can view tags applied to vector buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To view tags applied to a vector bucket using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **vector buckets**.
3. Choose the vector bucket name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section to view all of the tags applied to the vector bucket.
6. The **Tags** section shows the User-defined tags by default. You can select the AWS-generated tags tab to view tags applied to your vector bucket by AWS services.

Using the REST API

For information about the Amazon S3 REST API support for viewing the tags applied to a vector bucket, see the following section in the Amazon Simple Vectors API Reference:

ListTagsforResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to view tags applied to a vector bucket. To use the command replace the *`user input placeholders`* with your own information.

```
aws s3vectors list-tags-for-resource \
--resource-arn arn:aws:s3vectors:us-east-1:012345678900:bucket/acc-bucket
```

Deleting a tag from a vector bucket

You can remove tags from S3 vector buckets. An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 vector buckets. For more information about tagging vector buckets, see Using tags with S3 vector buckets.

> **Note**
>
> If you delete a tag and later learn that it was being used to track costs or for access control, you can add the tag back to the vector bucket.

Permissions

To delete a tag from a vector bucket, you must have the following permission:

- `s3vectors:UntagResource`

Troubleshooting errors

If you encounter an error when attempting to delete a tag from a vector bucket, you can do the following:

- Verify that you have the required Permissions to delete a tag from a vector bucket.

Steps

You can delete tags from vector buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To delete tags from a vector bucket using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **vector buckets**.
3. Choose the vector bucket name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and select the checkbox next to the tag or tags that you would like to delete.
6. Choose **Delete**.
7. The **Delete user-defined tags** pop-up appears and asks you to confirm the deletion of the tag or tags you selected.
8. Choose **Delete** to confirm.

Using the REST API

For information about the Amazon S3 REST API support for deleting tags from a vector bucket, see the following section in the *Amazon S3 Vectors API Reference*:

UntagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to delete tags from a vector bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

```
aws s3vectors untag-resource \
--resource-arn arn:aws:s3vectors:us-east-1:012345678900:bucket/acc-bucket \
--tag-keys CostCenter Department
```

Vector indexes

> **Note**
>
> Choose your vector index configuration parameters carefully. After you create a vector index, you can't update the vector index name, dimension, distance metric, or non-filterable metadata keys. To change any of these values, you must create a new vector index.

Vector indexes are resources within vector buckets that store and organize vector data for efficient similarity search operations. When you create a vector index, you specify the distance metric (`Cosine` or `Euclidean`), the number of dimensions that a vector should have, and optionally a list of metadata fields that you want to exclude from filtering during similarity queries.

For more information about vector index limits per bucket, vector limits per index, and dimension limits per vector, see Limitations and restrictions.

Each vector index has a unique Amazon Resource Name (ARN). The ARNs of vector indexes follow the following format:

```
arn:aws:s3vectors:region:account-id:bucket/bucket-name/index/index-name
```

Vector index naming requirements

- Vector index names must be unique within the vector bucket.
- Vector index names must be between 3 and 63 characters long.
- Valid characters are lowercase letters (a-z), numbers (0-9), hyphens (-), and dots (.).

- Vector index names must begin and end with a letter or number.

Dimension requirements

A dimension is the number of values in a vector. All vectors added to the index must have exactly this number of values.

- A dimension must be an integer between 1 and 4096.
- A larger dimension requires more storage space.

Distance metric options

Distance metric specifies how similarity between vectors is calculated. When creating vector embeddings, choose your embedding model's recommended distance metric for more accurate results.

- **Cosine** – Measures the cosine of the angle between vectors. Best for normalized vectors and when direction matters more than magnitude.
- **Euclidean** – Measures the straight-line distance between vectors. Best when both direction and magnitude are important.

Non-filterable metadata keys

Metadata keys allow you to attach additional information to your vectors as key-value pairs during storage and retrieval. By default, all metadata is filterable, so you can use it to filter query results. However, you can designate specific metadata keys as non-filterable when you want to store information with vectors without using it for filtering.

Unlike default metadata keys, these keys can't be used as query filters. Non-filterable metadata keys can be retrieved but can't be searched, queried, or filtered. You can only access it after finding the index.

Non-filterable metadata keys allow you to enrich vectors with additional context that you want to retrieve with search results but don't need for filtering. A common example of a non-filterable metadata key is when you embed text into vectors and want to include the original text itself as non-filterable metadata. This allows you to return the source text alongside vector search results without increasing your filterable metadata size limits. Other examples include storing creation

timestamps, source URLs, or descriptive information purely for reference. Non-filterable metadata keys can be accessed when retrieving vectors but, unlike default metadata keys, these keys can't be used as query filters.

Requirements for non-filterable metadata keys are as follows.

- Non-filterable metadata keys must be unique within the vector index.
- Non-filterable metadata keys must be 1 to 63 characters long.
- Non-filterable metadata keys can't be modified after the vector index is created.
- S3 Vectors support up to 10 non-filterable metadata keys per index.

For more information about non-filterable metadata keys, see Non-filterable metadata.

Index mode

Each vector index has an index mode, which determines the query capabilities the index supports. The modes are `ENHANCED` and `CLASSIC`. We recommend `ENHANCED` for almost all workloads, because it gives you higher recall on filtered queries and additional filter operators. For what each mode supports, see Metadata filtering.

A new vector index inherits the default index mode of the vector bucket that contains it. A vector bucket created on or after September 30, 2026 has a default index mode of `ENHANCED`, and indexes in it cannot use `CLASSIC`. A vector bucket created before that date has a default index mode of `CLASSIC`, so that indexes you already had are unchanged and indexes you create in it later behave consistently with them.

You can read an index's mode with GetIndex and change it with UpdateIndexMode. You can read a bucket's default with GetVectorBucket and change it with PutVectorBucketDefaultIndexMode, which applies to indexes you create afterward. To move an existing index, see Changing a vector index's mode.

Topics

- Creating a vector index in a vector bucket
- Changing a vector index's mode
- Listing vector indexes
- Deleting a vector index
- Using tags with S3 vector indexes

Creating a vector index in a vector bucket

> **Note**
>
> Choose your vector index configuration parameters carefully. After you create a vector index, you can't update the vector index name, dimension, distance metric, or non-filterable metadata keys. To change any of these values, you must create a new vector index.

A vector index is a resource within a vector bucket that stores and organizes vector data for efficient similarity search. When you create a vector index, you define the characteristics that all vectors in that index must share, such as the dimension, the distance metric used for similarity calculations, and optionally non-filterable metadata keys. You can also optionally configure dedicated encryption settings and tags for the vector index at the time of index creation. For more information about vector index naming requirements, dimension requirements, distance metric options, and non-filterable metadata keys, see Limitations and restrictions. For more information about setting encryption configuration for vector indexes, see Data protection and encryption in S3 Vectors. For more information about setting tags, see Using tags with S3 vector buckets.

Vector indexes must be created within an existing vector bucket and require specific configuration parameters that can't be modified after creation.

Using the S3 console

To create a vector index

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Vector buckets**.
3. In the list of vector buckets, choose the name of the bucket where you want to create a vector index.
4. Choose **Create vector index**.
5. For **Vector index name**, enter a name for your vector index.

   Vector index names must be unique within the vector bucket. Index name must be between 3 and 63 characters. Valid characters are lowercase letters (a-z), numbers (0-9), hyphens (-), and dots (.). For more information about the vector index naming requirements, see Limitations and restrictions.
6. For **Dimension**, enter the number of values in each vector.

> **Note**
>
> - The value for **Dimension** determines how many numerical values each vector will contain.
> - All vectors added to this index must have exactly this number of values.
> - Dimension must be between 1 and 4096.
> - A larger dimension requires more storage space.
> - Choose based on your embedding model's output dimensions.

For more information about the dimension requirements, see Limitations and restrictions.

7. For **Distance metric**, choose one of the following options:
   - **Cosine** – Measures the cosine of the angle between vectors. Best for normalized vectors and when direction matters more than magnitude
   - **Euclidean** – Measures the straight-line distance between vectors. Best when both direction and magnitude are important.
8. (Optional) Under **Non-filterable metadata**, configure metadata keys that will be stored but not used for filtering:

   To add non-filterable metadata keys:

   a. Choose **Add key**.

   b. Enter a key name (1-63 characters and unique within this vector index).

   c. Repeat to add additional keys (maximum 10 keys).

   > **Note**
   >
   > You can attach filterable metadata as key-value pairs to each vector when you insert vector data after you create a vector index. By default, all metadata keys that are attached to vectors are filterable and can be used as filters in a similarity query. Only metadata keys that are specified as non-filterable during vector index creation are excluded from filtering. For more information about metadata size limits per vector, including both total and filterable metadata constraints, see Limitations and restrictions.

9. Review your configuration carefully.

> **Note**
>
> These settings can't be changed after creation.

10. Under **Encryption**, choose **Specify encryption type**. You have the option to **Use bucket settings for encryption** or override the encryption settings for the vector index. If you override the bucket-level settings, you have the option to specify encryption type for the vector index as **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** or the **Server-side encryption with Amazon S3 managed keys (SSE-S3)**. For more information about setting encryption configuration for vector indexes, see Data protection and encryption in S3 Vectors.
11. Under **Tags (Optional)**, you can add tags as key-value pairs to help track and organize vector index costs using AWS Billing and Cost Management. Enter a **Key** and a **Value**. To add another tag, choose **Add Tag**. You can enter up to 50 tags for a vector index. For more information, see Using tags with S3 vector buckets.
12. Choose **Create vector index**.

Using the AWS CLI

To create a vector index in a vector bucket, use the following example commands and replace the `user input placeholders` with your own information.

Example 1: Creating a vector index with non-filterable metadata keys

```
aws s3vectors create-index \
  --vector-bucket-name "amzn-s3-demo-vector-bucket" \
  --index-name "idx" \
  --data-type "float32" \
  --dimension 1 \
  --distance-metric "cosine" \
  --metadata-configuration '{"nonFilterableMetadataKeys":["nonFilterableKey1"]}'
```

Example 2: Creating a vector index without non-filterable metadata keys

```
aws s3vectors create-index \
  --vector-bucket-name "amzn-s3-demo-vector-bucket" \
```

```
  --index-name "idx2" \
  --data-type "float32" \
  --dimension 4096 \
  --distance-metric "euclidean"
```

In addition, all metadata (both filterable and non-filterable) is retrieved the same way by using the `GetVectors`, `ListVectors`, or `QueryVectors` API operations. The following CLI command shows how to retrieve vectors with metadata (including non-filterable metadata).

Example request:

```
aws s3vectors get-vectors \
  --vector-bucket-name "amzn-s3-demo-vector-bucket" \
  --index-name "idx" \
  --keys '["vec1", "vec3"]' \
  --return-data \
  --return-metadata \
```

Example response:

```
{
    "vectors": [
        {
            "key": "vec1",
            "data": {
                "float32": [
                    0.10000000149011612,
                    0.20000000298023224,
                    0.30000001192092896,
                    0.4000000059604645,
                    0.5
                ]
            },
            "metadata": {
                "category": "test",
                "text": "First vector"
            }
        },
        {
            "key": "vec3",
            "data": {
```

```
                "float32": [
                    0.6000000238418579,
                    0.699999988079071,
                    0.800000011920929,
                    0.8999999761581421,
                    1.0
                ]
            },
            "metadata": {
                "text": "Third vector",
                "category": "test"
            }
        }
    ]
}
```

The response will include all metadata associated with the vector, regardless of whether it was specified as filterable or non-filterable during index creation.

Using the AWS SDKs

SDK for Python (Boto3)

```
import boto3

# Create a S3 Vectors client in the AWS Region of your choice.
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

#Create a vector index "movies" in the vector bucket "media-embeddings" without non-
filterable metadata keys
s3vectors.create_index(
    vectorBucketName="media-embeddings",
    indexName="movies",
    dimension=3,
    distanceMetric="cosine",
    dataType = "float32"
)


#Create a vector index "movies" in the vector bucket "media-embeddings" with non-
filterable metadata keys
s3vectors.create_index(
    vectorBucketName="media-embeddings",
```

```
    indexName="movies",
    dimension=3,
    distanceMetric="cosine",
    dataType = "float32",
    metadataConfiguration= {"nonFilterableMetadataKeys":
 ["nonFilterableMetadataKey1"]}
)
```

Changing a vector index's mode

A vector index whose index mode is ENHANCED applies your metadata filter before the vector search, ensuring high recall even when filters match a small fraction of vectors, and it supports the `$startsWith` filter operator. Indexes in vector buckets created before September 30, 2026 use CLASSIC by default, including indexes you create in those buckets later. We recommend moving these indexes to ENHANCED.

Changing an index to ENHANCED takes effect in place, with no re-ingestion, no changes to your queries or your application, and no additional charge. To change an index's mode, call UpdateIndexMode. To see an index's current mode, call GetIndex or view the index's detail page in the Amazon S3 console.

To change an index's mode (console)

1. Sign in to the console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Vector buckets**.
3. Choose the vector bucket that contains the index, then choose the index.
4. Choose **Enable enhanced index mode**.
5. Review the information in the confirmation dialog, select the acknowledgement, and then choose **Enable**.

```
aws s3vectors update-index-mode --vector-bucket-name "amzn-s3-demo-vector-bucket" --
index-name "idx" --index-mode "ENHANCED"
```

Before you change an index, confirm that your queries use 100 or fewer filter constraints. On an ENHANCED index, a query that exceeds the limit returns a validation error. For how constraints are

counted, and for how to consolidate or split a query that exceeds the limit, see Filter constraints per query. For how a filter affects query latency, see Query performance with filters.

Topics

- Test an index before you change it
- Set the index mode for new indexes in a vector bucket
- Turn off the enhanced index mode

Test an index before you change it

On a `CLASSIC` index, you can set the `queryMode` parameter to `ENHANCED` on a QueryVectors request to run that query with pre-filtering while the index stays as it is. You can also set `queryMode` to `ENHANCED` on an index whose index mode is already `ENHANCED`, where it has no additional effect.

If a query uses `$startsWith` under `queryMode=ENHANCED` and you later remove `queryMode`, the request returns a validation error, because `$startsWith` requires `ENHANCED` query behavior. Update the index mode to use `$startsWith` without setting `queryMode` on every request.

Set the index mode for new indexes in a vector bucket

Each vector bucket has a default index mode, inherited by new vector indexes. Setting the default applies to indexes you create afterward, and does not change any index that already exists in the bucket. To read a bucket's default, call GetVectorBucket, or view the bucket's **Properties** tab in the Amazon S3 console.

To move a vector bucket and its indexes to `ENHANCED`, complete the following steps.

1. Set the bucket's default index mode to `ENHANCED` with PutVectorBucketDefaultIndexMode, so that vector indexes you create afterward use `ENHANCED`.
2. List the vector indexes in the bucket with ListIndexes, and call GetIndex on each one to find those whose index mode is `CLASSIC`.
3. Call UpdateIndexMode with `indexMode` set to `ENHANCED` on each of those indexes.

```
aws s3vectors put-vector-bucket-default-index-mode --vector-bucket-name "amzn-s3-demo-vector-bucket" --default-index-mode "ENHANCED"
```

Turn off the enhanced index mode

Changing an index to ENHANCED is reversible if your vector bucket was created before September 30, 2026. You can change the index back to CLASSIC with the AWS CLI, the AWS SDKs, or the Amazon S3 REST API. You cannot do this from the Amazon S3 console.

```
aws s3vectors update-index-mode --vector-bucket-name "amzn-s3-demo-vector-bucket" --
index-name "idx" --index-mode "CLASSIC"
```

On a CLASSIC index, a query with a selective filter can return fewer than top K results, and $startsWith is available only when the request sets queryMode to ENHANCED. For which indexes accept CLASSIC, see UpdateIndexMode in the *Amazon S3 API Reference*.

Listing vector indexes

You can view all vector indexes within a vector bucket. The listing operation supports prefix-based filtering to help you find specific indexes when you have many indexes in a bucket. For more information about ListIndexes, prefix limits, and response limits, see ListIndexes in the Amazon Simple Storage Service API Reference.

Prefix search capability

Prefix search allows you to list indexes that start with a specific prefix, making it easier to organize and find related vector indexes. This is particularly useful when you use naming conventions that group related indexes together:

- **By data type:** text-embeddings-, image-features-, audio-vectors-
- **By model:** model1-embeddings-, model2-vectors-, custom-model-
- **By use case:** search-index-, recommendation-, similarity-
- **By environment:** prod-vectors-, staging-vectors-, dev-vectors-

Using the S3 console

To list vector indexes

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Vector buckets**.
3. In the list of vector buckets, choose the name of the bucket containing the indexes you want to view.
4. The console displays a comprehensive list of all vector indexes in the bucket, including:

   - **Name** – The name for each index.
   - **Create date** – When the index was created.
   - **Amazon Resource Name (ARN)** – Full ARN for each index.

To filter the list

1. Enter an index name or prefix in the search box above the index list. Use prefixes to find groups of related indexes.
2. The list updates in real-time as you type.

Using the AWS CLI

Use the following example commands and replace the *`user input placeholders`* with your own information.

To list indexes with a specific prefix in a vector bucket

Example request:

```
aws s3vectors list-indexes \
  --vector-bucket-name "amzn-s3-demo-bucket" \
  --prefix "idx" \
  --max-results 1
```

Example response:

```
{
    "nextToken":
 "lObb29ZkzxMGtBXs97Rkbs26xdtKemu4brsnq2jX8DCocADkILv5cRphemXS3PXXFnQBihQBmESgEeKaGA",
    "indexes": [
        {
```

```
            "vectorBucketName": "amzn-s3-demo-bucket",
            "indexName": "idx",
            "indexArn": "arn:aws:s3vectors:aws-region:111122223333:bucket/amzn-s3-demo-
vector-bucket/index/idx",
            "creationTime": "2025-06-12T15:50:23+00:00"
        }
    ]
}
```

To list indexes with pagination

Example request:

```
aws s3vectors list-indexes \
  --vector-bucket-name "amzn-s3-demo-bucket" \
  --prefix "idx" \
  --next-token
 "l0bb29ZkzxMGtBXs97Rkbs26xdtKemu4brsnq2jX8DCocADkILv5cRphemXS3PXXFnQBihQBmESgEeKaGA"
```

Example response:

```
{
    "indexes": [
        {
            "vectorBucketName": "amzn-s3-demo-bucket",
            "indexName": "idx2",
            "indexArn": "arn:aws:s3vectors:aws-region:111122223333:bucket/amzn-s3-demo-
vector-bucket/index/idx2",
            "creationTime": "2025-06-12T15:45:37+00:00"
        }
    ]
}
```

Using the AWS SDKs

SDK for Python (Boto3)

```
import boto3

# Create a S3 Vectors client in the AWS Region of your choice.
s3vectors = boto3.client("s3vectors", region_name="us-west-2")
```

```
#List vector indexes in your vector bucket
response = s3vectors.list_indexes(vectorBucketName="media-embeddings")
indexes = response["indexes"]
print(indexes)
```

Deleting a vector index

You can delete a vector index when you no longer need it. This operation permanently removes the index and all vectors that are stored within it.

> **Important**
>
> When you delete a vector index, you need to know the following:
>
> - You can delete vector indexes even when the indexes contain vectors.
> - All vectors stored in the index are permanently deleted
> - All metadata associated with those vectors is permanently lost
> - The operation can't be undone or reversed
> - Any ongoing operations on the index will fail immediately
> - Applications querying the index will receive errors
> - The index name becomes available for reuse within the bucket

Using the AWS CLI

Before you delete a vector index, verify the vector index. For more information about how to check the index details, see GetIndex in the *Amazon S3 API Reference*. For more information about how to list vectors inside the index to see what will be deleted, see Listing vector indexes.

To delete a vector index, use the following example commands. Replace the *`user input placeholders`* with your own information.

```
aws s3vectors delete-index --vector-bucket-name "amzn-s3-demo-vector-bucket" \
         --index-name "idx2"
```

For more information about how to verify whether the index is deleted, see Listing vector indexes.

Using the AWS SDKs

SDK for Python (Boto3)

```
import boto3

# Create a S3 Vectors client in the AWS Region of your choice.
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

#Delete a vector index
response = s3vectors.delete_index(
    vectorBucketName="media-embeddings",
    indexName="movies")
```

Using tags with S3 vector indexes

An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 vector indexes. You can tag S3 vector indexes when you create them or manage tags on existing vector indexes. For general information about tags, see Tagging for cost allocation or attribute-based access control (ABAC).

> **Note**
>
> There is no additional charge for using tags on vector indexes beyond the standard S3 API request rates. For more information, see Amazon S3 pricing.

Common ways to use tags with vector indexes

Use tags on your S3 vector indexes for:

- **Cost allocation** – Track storage costs by vector index tag in AWS Billing and Cost Management. For more information, see Using tags for cost allocation.
- **Attribute-based access control (ABAC)** – Scale access permissions and grant access to S3 vector indexes based on their tags. For more information, see Using tags for attribute-based access control (ABAC).

> **Note**
>
> You can use the same tags for both cost allocation and access control.

ABAC for S3 vector indexes

Amazon S3 vector indexes support attribute-based access control (ABAC) using tags. Use tag-based condition keys in your AWS organizations, IAM, and S3 vector index policies. For enterprises, ABAC inAmazon S3 supports authorization across multiple AWS accounts.

In your IAM policies, you can control access to S3 vector indexes based on the vector index's tags by using the following global condition keys:

`aws:ResourceTag/key-name`

Use this key to compare the tag key-value pair that you specify in the policy with the key-value pair attached to the resource. For example, you could require that access to a resource is allowed only if the resource has the attached tag key `Dept` with the value `Marketing`. For more information, see Controlling access to AWS resources.

`aws:RequestTag/key-name`

Use this key to compare the tag key-value pair that was passed in the request with the tag pair that you specify in the policy. For example, you could check whether the request includes the tag key `Dept` and that it has the value `Accounting`. For more information, see Controlling access during AWS requests. You can use this condition key to restrict which tag key-value pairs can be passed during the `TagResource` and `CreateIndex` API operations.

`aws:TagKeys`

Use this key to compare the tag keys in a request with the keys that you specify in the policy. We recommend that when you use policies to control access using tags, use the `aws:TagKeys` condition key to define what tag keys are allowed. For example policies and more information, see Controlling access based on tag keys. You can create an S3 vector index with tags. To allow tagging during the `CreateVectorBucket` API operation, you must create a policy that includes both the `s3vectors:TagResource` and `s3vectors:CreateVectorBucket` actions. You can then use the `aws:TagKeys` condition key to enforce using specific tags in the `CreateVectorBucket` request.

Example ABAC policies for vector indexes

See the following example ABAC policies for Amazon S3 vector indexes.

1.1 - IAM policy to create or modify vector indexes with specific tags

In this IAM policy, users or roles with this policy can only create S3 vector indexes if they tag the vector index with the tag key `project` and tag value `Trinity` in the vector index creation request. They can also add or modify tags on existing S3 vector indexes as long as the `TagResource` request includes the tag key-value pair `project:Trinity`. This policy does not grant read, write, or delete permissions on the vector indexes or its objects.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateVectorIndexWithTags",
      "Effect": "Allow",
      "Action": [
        "s3vectors:CreateIndex",
        "s3vectors:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/project": [
            "Trinity"
          ]
        }
      }
    }
  ]
}
```

1.2 - IAM policy to modify tags on existing resources maintaining tagging governance

In this IAM policy, IAM principals (users or roles) can modify tags on a vector index only if the value of the vector index's `project` tag matches the value of the principal's `project` tag. Only the four tags `project`, `environment`, `owner`, and `cost-center` specified in the `aws:TagKeys` condition keys are permitted for these vector indexes. This helps enforce tag governance, prevents unauthorized tag modifications, and keeps the tagging schema consistent across your vector indexes.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceTaggingRulesOnModification",
      "Effect": "Allow",
      "Action": [
        "s3vectors:TagResource"
      ],
      "Resource": "arn:aws:s3vectors:us-west-2:111122223333:bucket/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "project",
            "environment",
            "owner",
            "cost-center"
          ]
        }
      }
    }
  ]
}
```

Managing tags for vector indexes

You can add or manage tags for S3 vector indexes using the Amazon S3 Console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or using the S3 APIs: TagResource, UntagResource, and ListTagsForResource. For more information, see:

Topics

- Creating vector indexes with tags
- Adding a tag to a vector index
- Viewing vector index tags
- Deleting a tag from a vector index

Creating vector indexes with tags

You can tag Amazon S3 vector indexes when you create them. There is no additional charge for using tags on vector indexes beyond the standard S3 API request rates. For more information, see Amazon S3 pricing. For more information about tagging vector indexes, see Using tags with S3 vector indexes.

Permissions

To create a vector index with tags, you must have the following permissions:

- `s3vectors:CreateIndex`
- `s3vectors:TagResource`

Troubleshooting errors

If you encounter an error when attempting to create a vector index with tags, you can do the following:

- Verify that you have the required Permissions to create the vector index and add a tag to it.
- Check your IAM user policy for any attribute-based access control (ABAC) conditions. You may be required to label your vector indexes only with specific tag keys and values. For more information, see Using tags for attribute-based access control (ABAC).

Steps

You can create a vector index with tags applied by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To create a vector index with tags using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **vector indexes**.
3. Choose **create vector index** to create a new vector index.
4. Create a vector index as you normally would; see Creating a vector index in a vector bucket.

5. On the **Create vector index** page, **Tags** is an option when creating a new vector index.
6. Enter a name for the vector index.
7. Choose **Add new Tag** to open the Tags editor and enter a tag key-value pair. The tag key is required, but the value is optional.
8. To add another tag, select **Add new Tag** again. You can enter up to 50 tag key-value pairs.
9. After you complete specifying the options for your new vector index, choose **Create vector index**.

Using the REST API

For information about the Amazon S3 REST API support for creating a vector index with tags, see the following section in the *Amazon S3 Vectors API Reference*:

CreateIndex

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to create a vector index with tags by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

When you create a vector index you must provide configuration details and use the following naming convention: `example-vector-index`

```
aws s3vectors create-index --vector-bucket-name acc-bucket --data-type "float32" \
 --index-name accounts-index --dimension 1024 --distance-metric euclidean \
 --tags Department=Accounting,Stage=Prod
```

Adding a tag to a vector index

You can add tags to Amazon S3 vector indexes and modify these tags. There is no additional charge for using tags on vector indexes beyond the standard S3 API request rates. For more information, see Amazon S3 pricing. For more information about tagging vector indexes, see Using tags with S3 vector indexes.

Permissions

To add a tag to a vector index, you must have the following permission:

- `s3vectors:TagResource`

Troubleshooting errors

If you encounter an error when attempting to add a tag to a vector index, you can do the following:

- Verify that you have the required Permissions to add a tag to a vector index.
- If you attempted to add a tag key that starts with the AWS reserved prefix `aws:`, change the tag key and try again.

Steps

You can add tags to vector indexes by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWSSDKs.

Using the S3 console

To add tags to a vector index using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **vector indexes**.
3. Choose the vector index name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and choose **Add new Tag**.
6. This opens the **Add Tags** page. You can enter up to 50 tag key value pairs.
7. If you add a new tag with the same key name as an existing tag, the value of the new tag overrides the value of the existing tag.
8. You can also edit the values of existing tags on this page.
9. After you have added the tag(s), choose **Save changes**.

Using the REST API

For information about the Amazon S3 REST API support for adding tags to a vector index, see the following section in the *Amazon S3 Vectors API Reference*:

TagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to add tags to a vector index by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

```
aws s3vectors tag-resource \
--resource-arn arn:aws:s3vectors:us-east-1:012345678900:bucket/acc-bucket/
index/accounts-index \
--tags Stage=Prod,CostCenter=Marketing
```

Viewing vector index tags

You can view or list tags applied to Amazon S3 vector indexes. For more information about tagging vector indexes, see Using tags with S3 vector indexes.

Permissions

To view tags applied to a vector index, you must have the following permission:

- `s3vectors:ListTagsForResource`

Troubleshooting errors

If you encounter an error when attempting to list or view the tags of a vector index, you can do the following:

- Verify that you have the required Permissions to view or list the tags of the vector index.

Steps

You can view tags applied to vector indexes by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To view tags applied to a vector index using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **vector indexes**.
3. Choose the vector index name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section to view all of the tags applied to the vector index.
6. The **Tags** section shows the User-defined tags by default. You can select the AWS-generated tags tab to view tags applied to your vector index by AWS services.

Using the REST API

For information about the Amazon S3 REST API support for viewing the tags applied to a vector index, see the following section in the Amazon Simple Vectors API Reference:

ListTagsforResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to view tags applied to a vector index. To use the command replace the *`user input placeholders`* with your own information.

```
aws s3vectors list-tags-for-resource \
  --resource-arn arn:aws:s3vectors:us-east-1:012345678900:bucket/acc-bucket/
index/accounts-index
```

Deleting a tag from a vector index

You can remove tags from S3 vector indexes. An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 vector indexes. For more information about tagging vector indexes, see Using tags with S3 vector indexes.

> **Note**
>
> If you delete a tag and later learn that it was being used to track costs or for access control, you can add the tag back to the vector index.

Permissions

To delete a tag from a vector index, you must have the following permission:

- `s3vectors:UntagResource`

Troubleshooting errors

If you encounter an error when attempting to delete a tag from a vector index, you can do the following:

- Verify that you have the required Permissions to delete a tag from a vector index.

Steps

You can delete tags from vector indexes by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To delete tags from a vector index using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **vector indexes**.
3. Choose the vector index name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and select the checkbox next to the tag or tags that you would like to delete.
6. Choose **Delete**.
7. The **Delete user-defined tags** pop-up appears and asks you to confirm the deletion of the tag or tags you selected.
8. Choose **Delete** to confirm.

Using the REST API

For information about the Amazon S3 REST API support for deleting tags from a vector index, see the following section in the *Amazon S3 Vectors API Reference*:

UntagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to delete tags from a vector index by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

```
aws s3vectors untag-resource \
--resource-arn arn:aws:s3vectors:us-east-1:012345678900:bucket/acc-bucket/
index/accounts-index \
--tag-keys CostCenter Department
```

Vectors

Each vector consists of a key, which uniquely identifies each vector in a vector index. Additionally, you can attach metadata (for example, year, author, genre, location) as key value pairs to each vector.

Vector data operations include inserting, listing, querying, and deleting vectors. To generate new vector embeddings of your unstructured data, you can use the InvokeModel API operation from Amazon Bedrock to specify the model ID of the embedding model that you want to use. Additionally, the open-source Amazon S3 Vectors Embed CLI tool provides a simplified way to generate embeddings and perform semantic searches from the command line. For more information about this open source tool that automates both vector embedding generation with Amazon Bedrock foundation models and semantic search operations within your S3 vector indexes, see Creating vector embeddings and performing semantic searches with `s3vectors-embed-cli`.

Vector concepts

Vector keys: Each vector is identified by a unique vector key within the index. Vector keys can be up to 1,024 characters long and must be unique within the vector index. Keys are case-sensitive and can contain any UTF-8 characters.

Vector dimension: A dimension is the number of values in a vector. Larger dimensions require more storage space. All vectors in an index must have the same number of dimensions, which is specified when you create the index. A dimension must be an integer between 1 and 4096.

Metadata: You can attach metadata to vectors as key-value pairs to provide additional context and enable filtering during queries. Metadata includes both filterable and non-filterable metadata keys. Filterable metadata is used for query filtering. Non-filterable metadata keys are specified during

a vector index creation and provides additional context but can't be used for filtering. Metadata supports string, number, and boolean types. For more information about filterable and non-filterable metadata, see Metadata filtering. For more information about metadata limits, including size limits per vector and maximum metadata keys per vector, see Limitations and restrictions.

Topics

- Inserting vectors into a vector index
- Listing vectors
- Querying vectors
- Deleting vectors from a vector index
- Metadata filtering

Inserting vectors into a vector index

You can add vectors to a vector index with the PutVectors API operation. Each vector consists of a key, which uniquely identifies each vector in a vector index. If you put a vector with a key that already exists in the index, it will overwrite the existing vector completely, which makes the previous vector no longer searchable. To maximize write throughput and optimize for costs, it's recommended that you insert vectors in large batches, up to the maximum batch size for `PutVectors`. However, for workloads that need to use smaller batches - such as when live, incoming vector data must become immediately searchable - you can achieve higher write throughput by using a higher number of concurrent `PutVectors` requests, up to the maximum allowed requests per second limit. For more information about the maximum batch size for `PutVectors`, which is the limit of vectors per `PutVectors` API call, and the maximum requests and vectors per second limit, see Limitations and restrictions. Additionally, you can attach metadata (for example, year, author, genre, location) as key-value pairs to each vector. By default, all metadata keys that are attached to vectors are filterable and can be used as filters in a similarity query. Only metadata keys that are specified as non-filterable during vector index creation are excluded from filtering. S3 vector indexes support string, number, boolean, and list types of metadata. For more information about the total metadata size limit per vector and the filterable metadata size limit per vector, see Limitations and restrictions. If the metadata size exceeds these limits, the `PutVectors` API operation will return a `400 Bad Request` error.

Before adding vector data to your vector index with the `PutVectors` API operation, you need to convert your raw data into vector embeddings, which are numerical representations of your content as arrays of floating-point numbers. The vector embeddings capture the semantic meaning

of your content, enabling similarity searches once they're stored in your vector index through the `PutVectors` operation. You can generate vector embeddings using various methods depending on your data type and use case. These methods include using machine learning frameworks, specialized embedding libraries, or AWS services such as Amazon Bedrock. For example, if you're using Amazon Bedrock, you can generate embeddings with the InvokeModel API operation and your preferred embedding model.

Additionally, Amazon Bedrock Knowledge Bases provides a fully managed end-to-end RAG workflow where Amazon Bedrock automatically fetches data from your S3 data source, converts content into text blocks, generates embeddings, and stores them in your vector index. You can then query the knowledge base and generate responses based on chunks retrieved from your source data.

Furthermore, the open-source Amazon S3 Vectors Embed CLI tool provides a simplified way to generate embeddings and perform semantic searches from the command line. For more information about this open source tool that automates both vector embedding generation with Amazon Bedrock foundation models and semantic search operations within your S3 vector indexes, see Creating vector embeddings and performing semantic searches with `s3vectors-embed-cli`.

Note

When inserting vector data into your vector index, you must provide the vector data as `float32` (32-bit floating point) values. If you pass higher-precision values to an AWS SDK, S3 Vectors converts the values to 32-bit floating point before storing them, and GetVectors, ListVectors, and QueryVectors operations return the `float32` values. Different AWS SDKs may have different default numeric types, so ensure your vectors are properly formatted as `float32` values regardless of which SDK you're using. For example, in Python, use `numpy.float32` or explicitly cast your values.

Using the AWS SDKs

SDK for Python (Boto3)

```
# Populate a vector index with embeddings from Amazon Titan Text Embeddings V2.
import boto3
import json

# Create Bedrock Runtime and S3 Vectors clients in the AWS Region of your choice.
```

```
bedrock = boto3.client("bedrock-runtime", region_name="us-west-2")
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

# Texts to convert to embeddings.
texts = [
    "Star Wars: A farm boy joins rebels to fight an evil empire in space",
    "Jurassic Park: Scientists create dinosaurs in a theme park that goes wrong",
    "Finding Nemo: A father fish searches the ocean to find his lost son"
]

# Generate vector embeddings.
embeddings = []
for text in texts:
    response = bedrock.invoke_model(
        modelId="amazon.titan-embed-text-v2:0",
        body=json.dumps({"inputText": text})
    )

    # Extract embedding from response.
    response_body = json.loads(response["body"].read())
    embeddings.append(response_body["embedding"])

# Write embeddings into vector index with metadata.
s3vectors.put_vectors(
    vectorBucketName="media-embeddings",
    indexName="movies",
    vectors=[
        {
            "key": "Star Wars",
            "data": {"float32": embeddings[0]},
            "metadata": {"source_text": texts[0], "genre":"scifi"}
        },
        {
            "key": "Jurassic Park",
            "data": {"float32": embeddings[1]},
            "metadata": {"source_text": texts[1], "genre":"scifi"}
        },
        {
            "key": "Finding Nemo",
            "data": {"float32": embeddings[2]},
            "metadata": {"source_text": texts[2], "genre":"family"}
        }
    ]
```

```
)
```

Listing vectors

You can list vectors in a vector index with the ListVectors API operation. For more information about the maximum number of vectors that can be returned per page, see Limitations and restrictions. The response includes a pagination token when results are truncated. For more information about the response elements of `ListVectors`, see ListVectors in the *Amazon S3 API Reference*. You can also use `ListVectors` to export vector data from a specified vector index. `ListVectors` is strongly consistent. After a WRITE operation, you can immediately list vectors with all changes reflected.

Using the AWS CLI

To list vectors, use the following example commands. Replace the *`user input placeholders`* with your own information.

The `segment-count` and `segment-index` parameters allow you to partition your listing operations across multiple parallel requests. When you specify a `segment-count` value (such as 2), you divide the index into that many segments. The `segment-index` parameter (starting from 0) determines which segment to list. This approach helps improve performance when listing large vector indexes by enabling parallel processing. For more information about `segment-count` and `segment-index`, see ListVectors in the *Amazon S3 API Reference*.

To list all vectors in an index

Example request:

```
aws s3vectors list-vectors \
  --vector-bucket-name "amzn-s3-demo-vector-bucket" \
  --index-name "idx" \
  --segment-count 2 \
  --segment-index 0 \
  --return-data \
  --return-metadata
```

Example response:

```
{
    "vectors": [
```

```
        {
            "key": "vec3",
            "data": {
                "float32": [0.4000000059604645]
            },
            "metadata": {
                "nonFilterableKey": "val4",
                "filterableKey": "val2"
            }
        }
    ]
}
```

To list vectors with pagination

Example request:

```
aws s3vectors list-vectors \
  --vector-bucket-name "amzn-s3-demo-vector-bucket" \
  --index-name "idx" \
  --segment-count 2 \
  --segment-index 0 \
  --return-data \
  --return-metadata \
  --next-token
 "zWfh7e57H2jBfBtRRmC7OfMwl209G9dg3j2qM6kM4t0rps6ClYzJykgMOil9eGqU5nhf_gTq53IfoUdTnsg"
```

Example response:

```
{
    "vectors": [
        {
            "key": "vec1",
            "data": {
                "float32": [0.5]
            },
            "metadata": {
                "nonFilterableKey": "val2",
                "filterableKey": "val1"
            }
        }
    ]
}
```

Using the AWS SDKs

SDK for Python (Boto3)

Example: List vectors in a vector index

```
import boto3

# Create a S3 Vectors client in the AWS Region of your choice.
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

#List vectors in your vector index

response = s3vectors.list_vectors(
    vectorBucketName="media-embeddings",
    indexName="movies",
    maxResults = 600,
    returnData = True,
    returnMetadata = True
)

vectors = response["vectors"]

print(vectors)
```

Example: List all vectors in a vector index in parallel

```
import boto3

# Create a S3 Vectors client in the AWS Region of your choice.
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

#List vectors in the 1st half of vectors in the index.
response = s3vectors.list_vectors(
    vectorBucketName="media-embeddings",
    indexName="movies",
    segmentCount=2,
    segmentIndex=1,
    maxResults = 600,
    returnData = True,
    returnMetadata = True
)
```

```
vectors = response["vectors"]

#List vectors starting from the 2nd half of vectors in the index.
# This can be ran in parallel with the first `list_vectors` call.
response = s3vectors.list_vectors(
    vectorBucketName="media-embeddings",
    indexName="movies",
    segmentCount=2,
    segmentIndex=1,
    maxResults = 600,
    returnData = True,
    returnMetadata = True
)

vectors = response["vectors"]

print(vectors)
```

Querying vectors

You can run a similarity query with the QueryVectors API operation, where you specify the query vector, the number of relevant results to return (the top K nearest neighbors), and the index ARN. Additionally, you can use metadata filters in a query, to search only the vectors that match the filter. If you make a request to filter on a non-filterable metadata field, the request will return a `400 Bad Request` error. On `CLASSIC` indexes, you can set the optional `queryMode` parameter to run a single query with `ENHANCED` behavior without changing the index. If you don't specify `queryMode`, the query uses the index's own index mode. For more information about metadata filtering, see Metadata filtering.

In the response, the vector keys are returned by default. You can optionally include the distance and metadata in the response. When `topK` exceeds the page size, results are returned across multiple pages. The response includes a `nextToken` that you use to retrieve the next page. Continue making `QueryVectors` requests with the same `queryVector`, `topK`, `filter`, and `queryMode` parameters, passing the `nextToken` from each response, until `nextToken` is no longer present. Alternatively, you can use the built-in paginators in the AWS SDKs to automatically iterate through all pages without manually handling `nextToken`. Pagination tokens remain valid for several minutes after the query is executed. If a pagination token expires, re-issue the original query to start a new session. Writes to the vector index between page retrievals will not

be reflected in that query session. For specific limits on `topK` and page size, see Limitations and restrictions.

When generating the query vector, you should use the same vector embedding model that was used to generate the initial vectors that are stored in the vector index. For example, if you use the Amazon Titan Text Embeddings V2 model in Amazon Bedrock to generate vector embeddings of your documents, use the same embedding model to convert a question to a query vector. Additionally, Amazon Bedrock Knowledge Bases provides a fully managed end-to-end RAG workflow where Amazon Bedrock automatically fetches data from your S3 data source, converts content into text blocks, generates embeddings, and stores them in your vector index. You can then query the knowledge base and generate responses based on chunks retrieved from your source data. For more information about how to query vectors from an Amazon Bedrock knowledge base in the console, see (Optional) Integrate S3 Vectors with Amazon Bedrock Knowledge Bases.

Furthermore, the open-source Amazon S3 Vectors Embed CLI tool provides a simplified way to perform semantic searches from the command line. This open source tool streamlines the query process by handling both the vector embedding generation with Amazon Bedrock foundation models and executing semantic search operations against your S3 vector indexes. For more information about using this tool for querying your vector data, see Creating vector embeddings and performing semantic searches with `s3vectors-embed-cli`.

S3 Vectors delivers sub-second response times for cold queries, leveraging Amazon S3 elastic throughput to efficiently search across billions of vectors. This makes it highly cost-effective for workloads with infrequent queries. For warm queries, S3 Vectors can deliver response times as low as 100ms, benefiting workloads with repeated or frequent query patterns. When query results are returned across multiple pages, subsequent pages can be accessed immediately. On a filtered query, response time also varies with the work your filter does; see Query performance with filters.

For performing similarity queries for your vector embeddings, several factors can affect average recall performance, including the vector embedding model, the size of the vector dataset (the number of vectors and dimensions), and the distribution of queries. S3 Vectors delivers 90%+ average recall for most datasets. Average recall measures the quality of query results. A 90% average recall means that the response contains 90% of the actual closest vectors (ground truth) that are stored in the vector index relative to the query vector. On a filtered query, recall also depends on how selective your filter is and on the vector index's index mode. An index whose mode is `ENHANCED` applies your filter before the search and only searches vectors that match it, ensuring high recall even when few vectors match. If your filtered queries return fewer results than you

expect, check the index's mode with GetIndex, and see Changing a vector index's mode. However, because actual performance may vary depending on your specific use cases, we recommend conducting your own tests with representative data and queries to validate that S3 Vectors meet your recall requirements.

Using the AWS SDKs

SDK for Python (Boto3)

```
# Query a vector index with an embedding from Amazon Titan Text Embeddings V2.
import boto3
import json

# Create Bedrock Runtime and S3 Vectors clients in the AWS Region of your choice.
bedrock = boto3.client("bedrock-runtime", region_name="us-west-2")
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

# Query text to convert to an embedding.
input_text = "adventures in space"

# Generate the vector embedding.
response = bedrock.invoke_model(
    modelId="amazon.titan-embed-text-v2:0",
    body=json.dumps({"inputText": input_text})
)

# Extract embedding from response.
model_response = json.loads(response["body"].read())
embedding = model_response["embedding"]

# Query vector index.
response = s3vectors.query_vectors(
    vectorBucketName="media-embeddings",
    indexName="movies",
    queryVector={"float32": embedding},
    topK=3,
    returnDistance=True,
    returnMetadata=True
)
print(json.dumps(response["vectors"], indent=2))

# Query vector index with a metadata filter.
response = s3vectors.query_vectors(
```

```
    vectorBucketName="media-embeddings",
    indexName="movies",
    queryVector={"float32": embedding},
    topK=3,
    filter={"genre": "scifi"},
    returnDistance=True,
    returnMetadata=True
)
print(json.dumps(response["vectors"], indent=2))

# Paginated query with manual nextToken handling.
query_params = dict(
    vectorBucketName="media-embeddings",
    indexName="movies",
    queryVector={"float32": embedding},
    topK=500,
    returnDistance=True,
    returnMetadata=True,
)
response = s3vectors.query_vectors(**query_params)
for vector in response["vectors"]:
    print(vector)
while response.get("nextToken") is not None:
    response = s3vectors.query_vectors(**query_params,
 nextToken=response["nextToken"])
    for vector in response["vectors"]:
        print(vector)

# Use the built-in paginator to automatically iterate through all pages.
paginator = s3vectors.get_paginator("query_vectors")

for page in paginator.paginate(
    vectorBucketName="media-embeddings",
    indexName="movies",
    queryVector={"float32": embedding},
    topK=500,
    returnDistance=True,
    returnMetadata=True,
):
    for vector in page["vectors"]:
        print(vector)
```

Deleting vectors from a vector index

You can delete specific vectors from a vector index by specifying their vector keys using the DeleteVectors API. This operation is useful for removing outdated or incorrect data while preserving the rest of your vector data.

Using the AWS CLI

To delete vectors, use the following example commands. Replace the *user input placeholders* with your own information.

```
aws s3vectors delete-vectors \
 --vector-bucket-name "amzn-s3-demo-vector-bucket" \
 --index-name "idx" \
 --keys '["vec2","vec3"]'
```

Using the AWS SDKs

SDK for Python (Boto3)

```
import boto3

# Create a S3 Vectors client in the AWS Region of your choice.
s3vectors = boto3.client("s3vectors", region_name="us-west-2")

#Delete vectors in a vector index
response = s3vectors.delete_vectors(
    vectorBucketName="media-embeddings",
    indexName="movies",
    keys=["Star Wars", "Finding Nemo"])
```

Metadata filtering

Metadata filtering allows you to filter query results based on specific attributes attached to your vectors. You can use metadata filters with query operations to find vectors that match both similarity criteria and specific metadata conditions.

S3 Vectors supports two types of metadata: filterable metadata and non-filterable metadata. The key difference is that filterable metadata can be used in query filters but has stricter size

limitations, while non-filterable metadata can't be used in filters but can store larger amounts of data within its size limits. For more information about metadata limits, including size limits per vector and maximum metadata keys per vector, see Limitations and restrictions.

S3 Vectors evaluates your metadata filter before it runs vector search, a technique known as pre-filtering. S3 Vectors first identifies the vectors that match your filter, then only searches those vectors for the most similar ones, ensuring high recall even when filters match a small fraction of vectors. For example, if you search for similar movie embeddings and you filter by genre='mystery', S3 Vectors first identifies the vectors where the genre metadata is 'mystery', then only searches those for the closest matches to your query vector.

Pre-filtering applies to vector indexes whose index mode is `ENHANCED`. Vector buckets created on or after September 30, 2026 create `ENHANCED` indexes. In a vector bucket created before that date, indexes have the index mode `CLASSIC`.

S3 Vectors performs vector search and filter evaluation in tandem on a vector index whose index mode is `CLASSIC`. S3 Vectors searches through candidate vectors in the index to find the top K similar vectors while simultaneously validating if each candidate vector matches your metadata filter conditions. On a `CLASSIC` index, queries with filters may return fewer than top K results when the vector index contains very few matching results.

To move an existing index to `ENHANCED`, see Changing a vector index's mode.

Topics

- Filterable metadata
- Filter constraints per query
- Examples of valid filterable metadata
- Non-filterable metadata
- Query performance with filters

Filterable metadata

Filterable metadata allows you to filter query results based on specific metadata values. By default, all metadata fields are filterable in a similarity query unless explicitly specified as non-filterable during vector index creation. S3 Vectors supports string, number, boolean, and list types of metadata with a size limit per vector. The metadata type is ideal for attributes that you want to filter on, such as categories, timestamps, or status values.

If the metadata size exceeds the supported limits, the PutVectors API operation will return a `400 Bad Request` error. For more information about the filterable metadata size limit per vector, see Limitations and restrictions.

The following operations can be used with filterable metadata.

Operator	Valid Input Types	Description
`$eq`	String, Number, Boolean	Exact match comparison for single values. When comparing with an array metadata value, returns true if the input value matches any element in the array. For example, `{"category": {"$eq": "documentary"}}` would match a vector with metadata `"category": ["documentary", "romance"]` .
`$ne`	String, Number, Boolean	Not equal comparison
`$gt`	Number	Greater than comparison
`$gte`	Number	Greater than or equal comparison
`$lt`	Number	Less than comparison
`$lte`	Number	Less than or equal comparison
`$startsWith`	String	Prefix match. Matches values that begin with the specified string.
`$in`	Non-empty array of primitives	Match any value in array

Operator	Valid Input Types	Description
`$nin`	Non-empty array of primitives	Match none of the values in array
`$exists`	Boolean	Check if field exists
`$and`	Non-empty array of filters	Logical AND of multiple conditions
`$or`	Non-empty array of filters	Logical OR of multiple conditions

`$startsWith` requires `ENHANCED` query behavior. It is available on a vector index whose index mode is `ENHANCED`, and on a `CLASSIC` index when the request sets `queryMode` to `ENHANCED`.

Filter constraints per query

On a vector index whose index mode is `ENHANCED`, a single query filter can use up to 100 filter constraints. Each value the filter evaluates counts as one constraint: `{"category": "electronics"}` is one, `{"region": {"$in": ["us-east-1", "us-west-2", "eu-west-1"]}}` is three, and `{"$and": [{"category": "electronics"}, {"price": {"$lte": 500}}]}` is two. This limit only applies to `ENHANCED` indexes.

Typical filtered queries are well within 100 constraints. For guidance on writing filters that stay within the limit, see S3 Vectors best practices.

Examples of valid filterable metadata

Simple equality

```
{"genre": "documentary"}
```

This filter matches vectors where the genre metadata key equals "documentary". When you don't specify an operator, S3 Vectors automatically uses the $eq operator.

Explicit equality

```
// Example: Exact match
```

```
{"genre": {"$eq": "documentary"}}
```

```
// Example: Not equal to
{"genre": {"$ne": "drama"}}
```

Numeric comparison

```
{"year": {"$gt": 2019}}
```

```
{"year": {"$gte": 2020}}
```

```
{"year": {"$lt": 2020}}
```

```
{"year": {"$lte": 2020}}
```

Prefix match

```
{"s3_path": {"$startsWith": "/marketing/"}}
```

This filter matches vectors whose s3_path metadata begins with "/marketing/". Useful for hierarchical paths and URL prefixes.

Array operations

```
{"genre": {"$in": ["comedy", "documentary"]}}
```

```
{"genre": {"$nin": ["comedy", "documentary"]}}
```

Existence check

```
{"genre": {"$exists": true}}
```

The `$exists` filter matches vectors that have a "genre" metadata key, regardless of the value that's stored for that metadata key.

Logical operations

```
{"$and": [{"genre": {"$eq": "drama"}}, {"year": {"$gte": 2020}}]}
```

```
{"$or": [{"genre": {"$eq": "drama"}}, {"year": {"$gte": 2020}}]}
```

Price range (Multiple conditions on the same field)

```
{"price": {"$gte": 10, "$lte": 50}}
```

For more information about how to query vectors with metadata filtering, see Metadata filtering.

Non-filterable metadata

Non-filterable metadata can't be used in query filters but can store larger amounts of contextual data than filterable metadata. It's ideal for storing large text chunks, detailed descriptions, or other contextual information that doesn't need to be searchable but can be returned with query results. For example, you might store full document text, image descriptions, or detailed product specifications as non-filterable metadata.

Non-filterable metadata keys must be explicitly configured during vector index creation. Once a metadata key is designated as non-filterable during index creation, it can't be changed to filterable later. You can configure multiple metadata keys as non-filterable per vector index, with each metadata key name limited to 63 characters. For more information about the maximum number of non-filterable metadata keys that's allowed per vector index, see Limitations and restrictions.

While you can't filter on non-filterable metadata, you can retrieve it alongside query results using the `return-metadata` parameter. You can use non-filterable metadata for some use cases as follows.

- Use it to provide context for your application without parsing separate data sources.
- Store larger text chunks that would exceed the filterable metadata size limits.
- Include it in vector exports by using the ListVectors API operation.

For more information about configuring non-filterable metadata, see Creating a vector index in a vector bucket.

Query performance with filters

Because an ENHANCED index evaluates your filter before it searches, the work a filtered query does depends on the filter you send. Three things increase that work, and with it the query's latency: a

larger vector index, a filter that matches a larger share of the vectors in the index, and a filter with more constraints.

Because actual performance varies with your dataset size, dimensions, and filters, we recommend testing with representative data and queries. For guidance on writing filters, see S3 Vectors best practices.

Limitations and restrictions

Amazon S3 Vectors has certain limitations and restrictions that you should be aware of when planning your vector storage and search applications.

- Vector buckets per AWS Region in an account: 10,000
- Vector indexes per vector bucket: 10,000
- Vectors per vector index: Up to 2 billion
- Dimension value per vector: 1 to 4,096
- Total metadata per vector: Up to 40 KB (filterable + non-filterable)
- Total metadata keys per vector: Up to 50
- Filterable metadata per vector: Up to 2 KB
- Non-filterable metadata keys per vector index: Up to 10
- Combined PutVectors and DeleteVectors requests per second per vector index: Up to 1,000
- Combined vectors inserted and deleted per second per vector index: Up to 2,500
- Request payload size: Up to 20 MiB
- Vectors per PutVectors API call: Up to 500
- Vectors per DeleteVectors API call: Up to 500
- Vectors per GetVectors API call: Up to 100
- Top-K results per QueryVectors request: Up to 10,000
- Filter constraints per QueryVectors request on an `ENHANCED` index: Up to 100
- Results per page in a QueryVectors response: Up to 100
- Vectors listed per page in a ListVectors response: Up to 1,000
- Vector buckets listed per page in a ListVectorBuckets response: Up to 500.
- Vector indexes listed per page in a ListIndexes response: Up to 500.
- Segment count for parallel listing in a ListVectors API call: Up to 16

S3 Vectors best practices

Amazon S3 Vectors delivers purpose-built, cost-optimized vector storage for use by AI-enabled applications and semantic search of your content stored in Amazon S3. Designed to provide S3 level elasticity and durability for storing vector datasets with sub-second query performance for cold queries and as low as 100 milliseconds for warm queries, S3 Vectors is ideal for applications that need to build and grow vector indexes. With S3 Vectors you can use a dedicated set of API operations to store, access, and perform similarity queries on vector data without provisioning any infrastructure. For more information, see Working with S3 Vectors and vector buckets.

To ensure the maximum benefit from S3 Vectors, we recommend that you perform the following best practices.

Inserting and deleting vectors

Your application can achieve up to one thousand PutVectors or DeleteVectors requests per second per vector index, or can insert or delete up to two thousand five hundred vectors per second per vector index — whichever limit is reached first. If you exceed the request rates, you might receive a `429 TooManyRequestsException` error.

To optimize costs, we recommend inserting and deleting vectors in large batches, up to the maximum batch size of 500 vectors per API request. If your workload requires smaller batches, you can send concurrent requests up to the 1,000 requests per second limit. To reach maximum throughput of 2,500 vectors per second, you can send 5 batches per second with 500 vectors each, or 1,000 batches per second with an average of 2.5 vectors each.

Accessing and querying vectors in an S3 vector index

Your application can achieve hundreds of QueryVectors, GetVectors, or ListVectors requests per second per S3 vector index. If you exceed the request rates, you might receive a `429 TooManyRequestsException` error. We recommend you use a retry mechanism and configure your application to send fewer requests.

Scaling across vector indexes

To improve query performance per vector index, consider configuring your application to divide vectors across multiple vector indexes when possible. For example, if you have multi-tenant workloads and your application queries each tenant independently, consider storing each tenant's vectors in a separate vector index. For more information, see Vector indexes.

Implementing multi-tenancy with separate vector indexes

You can achieve multi-tenancy by organizing your vector data using a single vector index for each tenant. You can use IAM and bucket policies to restrict each tenant's access to only their designated vector index. This approach helps maintain data isolation and simplifies management by removing the need to create separate buckets for each tenant. For more information, see Identity and Access management in S3 Vectors.

Configuring non-filterable metadata fields for vector indexes

When creating a vector index, configure metadata fields that don't require filtering as non-filterable metadata keys. For example, store text chunks for vector embeddings as non-filterable metadata fields when you need them only for reference. For more information, see Non-filterable metadata.

Choosing an index mode

Use the `ENHANCED` index mode for workloads that filter on metadata. It applies your filter before the vector search, ensuring high recall even when filters match a small fraction of vectors, and it supports the `$startsWith` operator. Use `CLASSIC` only if your queries can neither be simplified to 100 or fewer filter constraints nor split across smaller queries; for both techniques, see Writing filters for lower latency. To run a single query with `ENHANCED` behavior without changing an index, set `queryMode` to `ENHANCED` on a `QueryVectors` request.

Writing filters for lower latency

A filtered query's latency grows with the work its filter does: with the size of the vector index, with the share of the index your filter matches, and with the number of constraints the filter evaluates. To keep filtered queries fast, prefer selective filters and use the fewest constraints that express what you need.

When a query would otherwise filter on a long list of values, attach a single metadata key that captures the group those values belong to, and filter on that one key instead. Filtering on a list uses one constraint per value; the following uses 3 constraints:

```
{"documentId": {"$in": ["doc-001", "doc-002", "doc-003"]}}
```

A user who can access 300 documents would need a 300-value list, which uses 300 constraints. Instead, attach a `caseId` to each vector when you write it, and filter on that single key, which uses one constraint:

```
{"caseId": "case-4471"}
```

If a query cannot be simplified this way, you can split its filter across smaller queries, run them in parallel, and merge the results by distance. Set `returnDistance` to `true` on each query so that every result includes its distance to merge on. For queries that must match a large share of your index, you can also divide the vectors across multiple vector indexes so that each query filters less data.

Creating vector embeddings and performing semantic searches with `s3vectors-embed-cli`

Amazon S3 Vectors provides a standalone open-source tool called Amazon S3 Vectors Embed CLI (`s3vectors-embed-cli`) that simplifies working with vector data by combining embedding generation with vector operations in single commands. This CLI tool helps you get started with S3 Vectors quickly by automating vector embedding generation with Amazon Bedrock foundation models and semantic search operations within your vector indexes.

The S3 Vectors Embed CLI provides two main commands for integrating Amazon Bedrock embedding models with S3 Vectors operations:

- `s3vectors-embed put`: Generate and insert vector embeddings into an vector index. Specifically, convert text and images into vector embeddings through Amazon Bedrock models and automatically store them in your vector index.
- `s3vectors-embed query`: Generate vector embeddings from your query input through Amazon Bedrock models and query vectors in your vector index.

The Amazon S3 Vectors Embed CLI is available in the Amazon Web Services - Labs GitHub repository. For detailed installation instructions, command parameters, examples, and best practices, see the Amazon S3 Vectors Embed CLI GitHub repository.

For the lower-level S3 Vectors API operations that provide more control and customization, see Amazon S3 Vectors in the *Amazon Simple Storage Service API Reference*.

Using S3 Vectors with other AWS services

S3 Vectors integrates with other AWS services to enhance your vector processing capabilities and provide comprehensive solutions for AI and machine learning workloads. These integrations allow you to leverage the cost-effective storage of S3 Vectors alongside the specialized capabilities of other AWS services.

Available integrations

S3 Vectors provides native integrations with the following AWS services:

- Amazon OpenSearch Service - You can export a snapshot of a vector index to Amazon OpenSearch Service for high queries per second (QPS) and low latency vector search. Additionally, Amazon OpenSearch Service adds Amazon S3 Vectors as a new low-cost engine for customers who want to optimize cost while continuing to use Amazon OpenSearch Service API operations for advanced search functionality including hybrid search, aggregations, advanced filtering, faceted search, and more.
- Amazon Bedrock Knowledge Bases - Use S3 Vectors as your vector store for Retrieval Augmented Generation (RAG) applications, reducing storage costs while maintaining query performance for knowledge base operations. You can access this integration through the Amazon Bedrock console or Amazon SageMaker AI Unified Studio.

Integration benefits

These integrations provide several key advantages:

- **Cost optimization**: Store large vector datasets cost-effectively in S3 Vectors while using specialized services for specific workloads, such as using Amazon OpenSearch for advanced search functionality.
- **Performance flexibility**: Choose the right integration for your performance requirements: S3 Vectors for lower throughput storage and sporadic querying, and other services for high-throughput, low-latency operations.
- **Workflow integration**: Seamlessly incorporate vector operations into existing AWS based AI and ML pipelines.
- **Simplified management**: Reduce operational complexity by using managed integrations rather than building custom solutions.

Topics

- Using S3 Vectors with OpenSearch Service
- Using S3 Vectors with Amazon Bedrock Knowledge Bases

Using S3 Vectors with OpenSearch Service

Amazon S3 Vectors integrates with OpenSearch to provide flexible vector storage and search capabilities. This integration allows you to optimize costs by storing vectors in S3 Vectors while leveraging the advanced search features of OpenSearch.

There are two integrations between S3 Vectors and OpenSearch. One is to export vector data from S3 Vectors to OpenSearch Serverless for high-performance search capabilities. The other uses S3 Vectors as a cost-effective storage engine within OpenSearch while maintaining access to OpenSearch functionality.

Export to OpenSearch Serverless

You can export vector indexes from S3 Vectors to OpenSearch Serverless collections for high-performance search operations, hybrid search, aggregations, advanced filtering, and faceted search. When you export, the data is copied to OpenSearch Serverless while remaining in S3 Vectors, which means you'll be paying for both services during this time.

When to use this integration

Consider exporting to OpenSearch Serverless from S3 Vectors when you need the following:

- **Hybrid search capabilities** that combine vector similarity with keyword search.
- **High query throughput** for demanding workloads.
- **Low-latency responses** for real-time applications that require millisecond response times.
- **Advanced analytics** that include aggregations, faceted search, and complex filtering.

Prerequisites

Before using S3 Vectors with OpenSearch, ensure you have the following:

- An existing S3 vector bucket with vector indexes containing your data.
- Appropriate IAM permissions for both S3 Vectors and OpenSearch Service.

- An understanding of your performance requirements to choose the appropriate integration method.

Getting started

Using the AWS Management Console

To export vector data to OpenSearch

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Vector buckets**.
3. In the list of vector buckets, choose the name of the bucket containing the vector data that you want to export.
4. For **Vector indexes**, choose the radio button next to the vector index that you want to export.
5. Choose **Advanced search export**, then choose **Export to OpenSearch**.

> **Important**
>
> - **Point-in-time export**: The export captures data up to the initiation of the export. If you make updates to your vector data during the ingestion to OpenSearch, not all updates will be reflected in OpenSearch.
> - **One-time operation**: This is a one-time export and will not stay in sync with your S3 Vectors data. You must manually re-export to capture any subsequent changes.

Then, to configure and manage the integration of S3 Vectors with Amazon OpenSearch Service, you'll work primarily through the OpenSearch console.

To view exports to OpenSearch

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Vector buckets**.
3. In the list of vector buckets, choose the name of the bucket containing the vector data that you want to export.

4. For **Vector indexes**, choose **Advanced search export**, then choose **View exports to OpenSearch**.

Then, you'll view exports to OpenSearch through the OpenSearch console.

For more information about creating and managing OpenSearch Serverless collections, see Creating and managing Amazon OpenSearch Service Serverless collections in the *Amazon OpenSearch Service Developer Guide*.

OpenSearch with S3 Vectors engine

You can use S3 Vectors as the underlying storage engine for Amazon OpenSearch Managed clusters, providing cost-optimized vector storage while maintaining OpenSearch functionality.

When to use this integration

Consider using OpenSearch with the S3 Vectors engine when you need the following:

- **Hybrid search capabilities** that combine vector similarity with keyword search.
- **Lower query throughput** that may have less frequent or sporadic usage patterns.
- **Higher latency tolerance** for applications that can accept longer response times in exchange for cost savings.
- **Advanced analytics** that include aggregations, faceted search, and complex filtering.
- **Existing OpenSearch workflows** that you want to extend with cost-effective vector storage.

Prerequisites

Before using OpenSearch with S3 Vectors engine, ensure you have:

- An existing OpenSearch managed domain. For more information, see Creating and managing Amazon OpenSearch Service domains in the *Amazon OpenSearch Service Developer Guide*.
- An understanding of your performance requirements to choose the appropriate integration method.

Getting Started

To use OpenSearch with S3 Vectors engine, set your engine to `S3_Vectors` during index creation in OpenSearch. For more information about the template that you use when creating an index

in OpenSearch Service, including where to specify the engine type, see Methods and engines. For more information about the integration between OpenSearch and S3 Vectors engine, see Advanced search capabilities with an S3 Vectors engine in the *Amazon OpenSearch Service Developer Guide*.

Using S3 Vectors with Amazon Bedrock Knowledge Bases

S3 Vectors integrates with Amazon Bedrock Knowledge Bases and Amazon SageMaker AI Unified Studio to simplify and reduce the cost of vector storage for retrieval augmented generation (RAG) applications.

For more information about high-level CLI commands that integrate Amazon Bedrock embedding models with S3 Vectors operations, see Creating vector embeddings and performing semantic searches with `s3vectors-embed-cli`.

Topics

- Integration overview
- When to use this integration
- Supported embedding models
- Prerequisites and permissions
- Creating a knowledge base with S3 Vectors
- Managing and querying your knowledge base
- Limitations

Integration overview

When creating a knowledge base in Amazon Bedrock, you can select S3 Vectors as your vector store. This integration provides the following:

- **Cost savings** for RAG applications with large vector datasets.
- **Seamless integration** with the fully managed RAG workflow of Amazon Bedrock.
- **Automatic vector management** handled by the Amazon Bedrock service.
- **Sub-second cold query latency and as low as 100 millisecond warm query latency** for knowledge base retrieval operations.

Amazon Bedrock Knowledge Bases provides a fully managed end-to-end RAG workflow. When you create a knowledge base with S3 Vectors, Amazon Bedrock automatically fetches data from your S3 data source, converts content into text blocks, generates embeddings, and stores them in your vector index. You can then query the knowledge base and generate responses based on chunks retrieved from your source data.

When to use this integration

Consider using S3 Vectors with Amazon Bedrock Knowledge Bases when you need the following:

- **Cost-effective vector storage** for large datasets where sub-second query latency meets your application requirements.
- **Text and image-based document retrieval** for use cases like searching through manuals, policies, and visual content.
- **RAG applications** that prioritize storage cost optimization over ultra-low latency responses.
- **Managed vector operations** without needing to learn S3 Vectors API operations directly - you can continue using familiar Amazon Bedrock interfaces.
- **Long-term vector storage** with the durability and scalability of Amazon S3

This integration is ideal for organizations building RAG applications that need to search through and extract insights from written content and images, where the cost benefits of S3 Vectors align with acceptable query performance requirements.

Supported embedding models

Refer to supported models for vector embeddings in the *Amazon Bedrock User Guide*.

Prerequisites and permissions

Before creating a knowledge base with S3 Vectors, ensure you have the following:

- Appropriate IAM permissions for both S3 Vectors and Amazon Bedrock services. For more information about IAM permissions for S3 Vectors, see the section called “Identity and Access management in S3 Vectors”. For more information about IAM permissions for your Amazon Bedrock Knowledge Bases service role to access S3 Vectors, see Permissions to access your vector store in Amazon S3 Vectors in the *Amazon Bedrock User Guide*.
- Your source documents prepared for ingestion into the knowledge base.
- An understanding of your embedding model requirements.

When setting up security configurations, you can choose an IAM role that provides Amazon Bedrock permission to access required AWS services. You can let Amazon Bedrock create the service role or use your own custom role. If you use a custom role, configure a vector bucket policy that restricts access to the vector bucket and vector index to the custom role.

For detailed information about required permissions and IAM roles, see Create a service role for Amazon Bedrock Knowledge Bases in the *Amazon Bedrock User Guide*. The service role must also have permissions for S3 Vectors and AWS KMS API operations.

Creating a knowledge base with S3 Vectors

You can create a knowledge base that uses S3 Vectors through two methods.

Method one: Using the Amazon Bedrock console

When creating a knowledge base in the Amazon Bedrock console, you can choose "S3 vector bucket" as your vector store option. You have two setup options:

- **Quick create a new vector store** - Amazon Bedrock creates an S3 vector bucket and vector index and configures them with the required settings for you. By default, the vector bucket is encrypted using server-side encryption with Amazon S3 managed keys (SSE-S3). You can optionally encrypt the bucket using AWS KMS. For more information about **Quick create a new vector store** in the console, see Create a knowledge base by connecting to a data source in Amazon Bedrock Knowledge Bases in the *Amazon Bedrock User Guide*.
- **Choose a vector store you have created** - Choose an existing S3 vector bucket and vector index from your account that you've previously created. For more information about creating an S3 vector bucket and vector index in the Amazon Bedrock Knowledge Bases console, see the S3 Vectors tab in Prerequisites for using a vector store you created for a knowledge base in the *Amazon Bedrock User Guide*.

For detailed step-by-step instructions, see Create a knowledge base by connecting to a data source in Amazon Bedrock Knowledge Bases in the *Amazon Bedrock User Guide*.

Method two: Using Amazon SageMaker Unified Studio

You can also create and manage knowledge bases with S3 Vectors through Amazon Bedrock in Amazon SageMaker AI Unified Studio. This provides a unified development environment for building and testing AI applications that use knowledge bases.

Amazon Bedrock in SageMaker AI Unified Studio is designed for users who need integrated notebook capabilities and work across multiple AWS ML and analytics services. You can quickly create an S3 vector bucket and configure it as the vector store for your knowledge bases when you build generative AI applications.

For information about using S3 Vectors with Amazon Bedrock in SageMaker AI Unified Studio, see Add a data source to your Amazon Bedrock app in the *SageMaker AI Unified Studio User Guide*.

Managing and querying your knowledge base

Data synchronization and management

Amazon Bedrock Knowledge Bases offers ingestion job operations to keep your data source and vector embeddings synchronized. When you sync your data source, Amazon Bedrock scans each document and verifies whether it has been indexed into the vector store. You can also directly index documents into the vector store using the IngestKnowledgeBaseDocuments operation. Best practice is to create a separate vector store for each knowledge base to ensure data synchronization.

When you delete a knowledge base or data source resource, Amazon Bedrock offers two data deletion policies: `Delete` (default) and `Retain`. If you choose the `Delete` policy, vectors in the vector index and vector bucket are automatically deleted.

Querying and retrieval

After your knowledge base is set up, you can do the following:

- **Retrieve chunks** from your source data using the Retrieve API operation.
- **Generate responses** based on retrieved chunks using the RetrieveAndGenerate API operation.
- **Test queries** directly in the Amazon Bedrock console.

Responses are returned with citations to the original source data.

Limitations

When using S3 Vectors with Amazon Bedrock Knowledge Bases, you should know the following limitations:

- **Semantic search only**: S3 Vectors supports semantic search but not hybrid search capabilities.

- **S3 Vectors size limits**: Each vector has a total metadata size limit and a size limit for filterable metadata, which may limit custom metadata and filtering options. When using S3 Vectors as your vector store with Amazon Bedrock Knowledge Bases, you can attach up to 1KB of custom metadata and 35 metadata keys per vector.
- **Chunking strategy constraints**: When using very high token counts with hierarchical chunking in Amazon Bedrock Knowledge Bases, you can exceed the maximum metadata size limits as parent-child chunk relationships and hierarchical context are stored as non-filterable metadata in S3 Vectors. For more information about metadata size limits per vector, see Limitations and restrictions.
- **Floating-point vectors only**: Binary vector embeddings aren't supported.

For comprehensive guidance on working with Amazon Bedrock Knowledge Bases, see Retrieve data and generate AI responses with Amazon Bedrock Knowledge Bases in the *Amazon Bedrock User Guide*.

AWS Regions, endpoints, and quotas for S3 Vectors

The following sections include the supported AWS Regions and service quotas for Amazon S3 Vectors.

S3 Vectors AWS Regions and endpoints

Amazon S3 Vectors is currently available in the following AWS Regions. To connect programmatically to an AWS service, you use an endpoint. For more information about AWS service endpoints, see AWS service endpoints in the AWS General Reference. For more Amazon S3 endpoint information, see Amazon S3 endpoints and quotas in the AWS General Reference.

S3 Vectors endpoints are dual-stack endpoints that support requests over IPv6 and IPv4.

S3 Vectors supports private connectivity with AWS PrivateLink interface endpoints. For more information, see VPC endpoints for S3 Vectors.

S3 Vectors AWS Regions and endpoints

Regio Name	Region	Endpoint	Protocol	Signature Version(s) Support
Africa (Cape Town)	af-south-1	s3vectors.af-south-1.api.aws	HTTPS	4
Asia Pacific (Hong Kong)	ap-east-1	s3vectors.ap-east-1.api.aws	HTTPS	4
Asia Pacific (Hyder d)	ap-south-2	s3vectors.ap-south-2.api.aws	HTTPS	4
Asia Pacific (Jakar	ap-southeast-3	s3vectors.ap-southeast-3.api.aws	HTTPS	4
Asia Pacific (Malay)	ap-southeast-5	s3vectors.ap-southeast-5.api.aws	HTTPS	4
Asia Pacific (Melbo e)	ap-southeast-4	s3vectors.ap-southeast-4.api.aws	HTTPS	4
Asia Pacific (Mumb	ap-south-1	s3vectors.ap-south-1.api.aws	HTTPS	4

Regio Name	Region	Endpoint	Protocol	Signature Version(s) Support
Asia Pacific (New Zealar	ap-southeast-6	s3vectors.ap-southeast-6.api.aws	HTTPS	4
Asia Pacific (Osaka	ap-northeast-3	s3vectors.ap-northeast-3.api.aws	HTTPS	4
Asia Pacific (Seoul	ap-northeast-2	s3vectors.ap-northeast-2.api.aws	HTTPS	4
Asia Pacific (Singa e)	ap-southeast-1	s3vectors.ap-southeast-1.api.aws	HTTPS	4
Asia Pacific (Sydne	ap-southeast-2	s3vectors.ap-southeast-2.api.aws	HTTPS	4
Asia Pacific (Taipe	ap-east-2	s3vectors.ap-east-2.api.aws	HTTPS	4
Asia Pacific (Thaila)	ap-southeast-7	s3vectors.ap-southeast-7.api.aws	HTTPS	4
Asia Pacific (Tokyo	ap-northeast-1	s3vectors.ap-northeast-1.api.aws	HTTPS	4

Regio Name	Region	Endpoint	Protocol	Signature Version(s) Support
AWS Europ Sovere Cloud (Germ	eusc-de-east-1	s3vectors.eusc-de-east-1.api.amazonwebservices.eu	HTTPS	4
AWS GovCl (US-East)	us-gov-east-1	s3vectors.us-gov-east-1.api.aws	HTTPS	4
AWS GovCl (US-West)	us-gov-west-1	s3vectors.us-gov-west-1.api.aws	HTTPS	4
Canad (Centr	ca-central-1	s3vectors.ca-central-1.api.aws	HTTPS	4
Canad West (Calga	ca-west-1	s3vectors.ca-west-1.api.aws	HTTPS	4
Europ (Frank t)	eu-central-1	s3vectors.eu-central-1.api.aws	HTTPS	4
Europ (Irelan	eu-west-1	s3vectors.eu-west-1.api.aws	HTTPS	4
Europ (Lond	eu-west-2	s3vectors.eu-west-2.api.aws	HTTPS	4
Europ (Milan	eu-south-1	s3vectors.eu-south-1.api.aws	HTTPS	4

Regio Name	Region	Endpoint	Protocol	Signature Version(s) Support
Europ (Paris)	eu-west-3	s3vectors.eu-west-3.api.aws	HTTPS	4
Europ (Spain	eu-south-2	s3vectors.eu-south-2.api.aws	HTTPS	4
Europ (Stock m)	eu-north-1	s3vectors.eu-north-1.api.aws	HTTPS	4
Europ (Zurich	eu-central-2	s3vectors.eu-central-2.api.aws	HTTPS	4
Mexic (Centr	mx-central-1	s3vectors.mx-central-1.api.aws	HTTPS	4
South Ameri (Sao Paulo)	sa-east-1	s3vectors.sa-east-1.api.aws	HTTPS	4
US East (N. Virgini	us-east-1	s3vectors.us-east-1.api.aws	HTTPS	4
US East (Ohio)	us-east-2	s3vectors.us-east-2.api.aws	HTTPS	4
US West (N. Califor a)	us-west-1	s3vectors.us-west-1.api.aws	HTTPS	4

Regio Name	Region	Endpoint	Protocol	Signature Version(s) Support
US West (Oregc	us-west-2	s3vectors.us-west-2.api.aws	HTTPS	4

S3 Vectors service quotas

Quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account. The limits are the service quotas for S3 Vectors resources.

For detailed limits that apply to S3 Vectors, see Limitations and restrictions.

Unless otherwise noted, each quota is per AWS Region. You can request increases for some quotas, and other quotas can't be increased. For more information about requesting a quota increase, see Requesting a quota increase in the *Service Quotas User Guide*. For more Amazon S3 quota information, see Amazon S3 endpoints and quotas in the *AWS General Reference*.

Security in S3 Vectors

Amazon S3 Vectors offers highly durable vector storage. Data written to S3 Vectors is stored on S3, which is designed for 99.999999999% (11 nines) of data durability.

S3 Vectors implements a security approach that includes encryption at rest and secure data transmission. Whether you're storing millions of vectors for machine learning applications or building semantic search systems, S3 Vectors provides the security controls necessary to meet your organization's compliance and data protection requirements.

Topics

- Identity and Access management in S3 Vectors
- Data protection and encryption in S3 Vectors
- VPC endpoints for S3 Vectors

Identity and Access management in S3 Vectors

Access management in S3 Vectors follows AWS security best practices, providing multiple layers of control to ensure that only authorized users and applications can access your vector data. The service integrates with IAM and supports both identity-based and resource-based policies, giving you flexibility in how you structure and manage permissions across your organization.

Authenticating and authorizing requests

S3 Vectors uses AWS standard authentication and authorization mechanisms to secure access to vector buckets and their contents. Every request to S3 Vectors must be authenticated using valid AWS credentials, and the service evaluates permissions based on the combination of identity-based policies, resource-based policies, and any applicable service control policies.

The authentication process begins when a client makes a request to S3 Vectors using AWS credentials (access keys, temporary credentials from AWS STS, or IAM roles). The service validates these credentials and then evaluates the permissions associated with the authenticated identity against the requested action and target resource. This evaluation process considers multiple policy types and applies the principle of least privilege to determine whether the request should be allowed or denied.

Authorization in S3 Vectors operates at multiple levels of granularity. You can control access at the vector bucket level, individual vector index level, or even specific operations within an index. This hierarchical permission model allows you to implement sophisticated access control schemes that align with your organizational structure and data governance requirements.

Resource types defined for vector buckets

S3 Vectors defines specific resource types that can be referenced in IAM policies and resource-based policies. Understanding these resource types is essential for creating effective access control policies that provide the right level of access to the right users and applications.

The following table describes the resource types available in S3 Vectors.

Resource types available in S3 Vectors

Resource type	ARN format	Description
VectorBucket	arn:aws:s3vectors:*region*:*123456789012* :bucket/*bucket-name*	Represents a vector bucket and is used for bucket-level operations such as creating,

Resource type	ARN format	Description
		deleting, or configuring the bucket
Index	arn:aws:s3vectors:*region*:*123456789012* :bucket/*bucket-name* /index/*index-name*	Represents a vector index within a bucket and is used for index-specific operations such as querying vectors or managing index contents

Policy actions for vector buckets

S3 Vectors provides a comprehensive set of policy actions that correspond to the various operations you can perform on vector buckets and indexes. These actions are designed to provide fine-grained control over who can perform specific operations, allowing you to implement the principle of least privilege effectively.

The following table lists all available policy actions for S3 Vectors resources.

Policy actions for S3 Vectors resources

Resource type	API operations	Policy actions	Description of policy actions	Access level	Condition keys
Account	ListVectorBuckets	s3vectors:ListVectorBuckets	Grants permission to list all vector buckets in the account and region	List	
VectorBucket	CreateVectorBucket	s3vectors:CreateVectorBucket	Grants permission to create a new vector bucket with	Write	s3vectors:sseType, s3vectors:kmsKeyArn

Resource type	API operations	Policy actions	Description of policy actions	Access level	Condition keys
			specified configuration		
VectorBucket	GetVectorBucket	s3vectors:GetVectorBucket	Grants permission to retrieve vector bucket attributes and configuration	Read	
VectorBucket	PutVectorBucketDefaultIndexMode	s3vectors:PutVectorBucketDefaultIndexMode	Grants permission to set the default index mode for vector indexes created in a vector bucket	Write	
VectorBucket	DeleteVectorBucket	s3vectors:DeleteVectorBucket	Grants permission to delete an empty vector bucket	Write	
VectorBucket	ListIndexes	s3vectors:ListIndexes	Grants permission to list all indexes within a vector bucket	List	

Resource type	API operations	Policy actions	Description of policy actions	Access level	Condition keys
VectorBucket	PutVectorBucketPolicy	s3vectors:PutVectorBucketPolicy	Grants permission to apply or update a resource-based policy on a vector bucket	Permissions management	
VectorBucket	GetVectorBucketPolicy	s3vectors:GetVectorBucketPolicy	Grants permission to retrieve the resource-based policy attached to a vector bucket	Read	
VectorBucket	DeleteVectorBucketPolicy	s3vectors:DeleteVectorBucketPolicy	Grants permission to remove the resource-based policy from a vector bucket	Permissions management	

Resource type	API operations	Policy actions	Description of policy actions	Access level	Condition keys
Index	CreateIndex	s3vectors:CreateIndex	Grants permission to create a new vector index with specified dimensions and metadata configuration	Write	
Index	GetIndex	s3vectors:GetIndex	Grants permission to retrieve vector index attributes and configuration	Read	
Index	UpdateIndexMode	s3vectors:UpdateIndexMode	Grants permission to change the index mode of a vector index	Write	
Index	DeleteIndex	s3vectors:DeleteIndex	Grants permission to delete a vector index and all its contents	Write	

Resource type	API operations	Policy actions	Description of policy actions	Access level	Condition keys
Index	QueryVectors	(Required) s3vectors:QueryVectors	Grants permission to perform similarity queries against vectors in an index. **With `s3vectors:QueryVectors` only**, you can retrieve vector keys of approximate nearest neighbors and their computed distances from the query vector. This permission is sufficient only when you don't set any metadata filters and don't request	Read	

Resource type	API operations	Policy actions	Description of policy actions	Access level	Condition keys
			metadata (by keeping the `returnMetadata` parameter set to false or not specified).		

Resource type	API operations	Policy actions	Description of policy actions	Access level	Condition keys
		(Conditionally required): s3vectors:GetVectors	Required if you set metadata filters, set `returnMetadata` to true in your request. **With both `s3vectors:QueryVectors` and `s3vectors:GetVectors`**, you can filter results by using metadata criteria and retrieve vector keys along with their associated data, metadata, and computed distances from the query vector.	Read	

Resource type	API operations	Policy actions	Description of policy actions	Access level	Condition keys
Index	PutVectors	s3vectors:PutVectors	Grants permission to add or update vectors in an index	Write	
Index	GetVectors	s3vectors:GetVectors	Grants permission to retrieve specific vectors and their metadata by vector key	Read	

Resource type	API operations	Policy actions	Description of policy actions	Access level	Condition keys
Index	ListVectors	(Required) s3vectors:ListVectors	Grants permission to list vector keys in an index. **With `s3vectors:ListVectors` only**, you can list vector keys when both `returnData` and `returnMetadata` parameters are false or not specified.	Read	

Resource type	API operations	Policy actions	Description of policy actions	Access level	Condition keys
		(Conditionally required): s3vectors:GetVectors	Required if you set either `returnData` or `returnMetadata` parameter to true in your request. **With both `s3vectors:ListVectors` and `s3vectors:GetVectors`**, you can retrieve vector keys along with their associated data and metadata by setting `returnData` and `returnMetadata` to true.	Read	

Resource type	API operations	Policy actions	Description of policy actions	Access level	Condition keys
Index	DeleteVectors	s3vectors:DeleteVectors	Grants permission to delete specific vectors from an index	Write	

These actions can be combined in various ways to create policies that match your specific access requirements. For example, you might create a read-only policy that includes `s3vectors:GetVectorBucket`, `s3vectors:ListIndexes`, `s3vectors:QueryVectors`, and `s3vectors:GetVectors` actions, or a policy that includes query and vector retrieval permissions but excludes administrative actions like creating or deleting indexes.

Condition keys for vector buckets

Condition keys for vector buckets

	Condition keys	Description	Type
1	s3vectors:sseType	Filters access by server-side encryption type **Valid values:** `AES256 | aws:kms`	String
2	s3vectors:kmsKeyArn	Filters access by the AWS AWS KMS key ARN for the key used to encrypt a vector bucket	ARN

S3 Vectors identity-based policy examples

IAM identity-based policies are JSON documents that you attach to IAM users, groups, or roles to define what actions they can perform on S3 Vectors resources. These policies are evaluated in the context of the identity making the request and provide a centralized way to manage permissions across your AWS environment. Identity-based policies provide a clear audit trail of who has what permissions and can be easily modified as your access requirements evolve.

When designing identity-based policies for S3 Vectors, consider the different types of users and applications that will interact with your vector data. Common patterns include data scientists who need to query vectors, data engineers who need to load and manage vector data, administrators who need full control over bucket configuration, and applications that need specific read or write access to particular vector indexes.

Example policies

Administrative access policy

This policy provides full administrative access to S3 Vectors resources, suitable for platform administrators or DevOps teams:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAdministrativeAccess",
            "Effect": "Allow",
            "Action": [
                "s3vectors:CreateVectorBucket",
                "s3vectors:PutVectorBucketPolicy",
                "s3vectors:DeleteVectorBucket",
                "s3vectors:DeleteVectorBucketPolicy",
                "s3vectors:GetVectorBucket",
                "s3vectors:GetVectorBucketPolicy",
                "s3vectors:ListVectorBuckets",
                "s3vectors:CreateIndex",
                "s3vectors:DeleteIndex",
                "s3vectors:GetIndex",
                "s3vectors:ListIndexes",
                "s3vectors:DeleteVectors",
                "s3vectors:GetVectors",
                "s3vectors:ListVectors",
```

```
                "s3vectors:PutVectors",
                "s3vectors:QueryVectors"
            ],
            "Resource": "*"
        }
    ]
}
```

Application-specific access policy

This policy is designed for applications that need to perform specific operations on designated vector indexes:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowApplicationVectorAccess",
            "Effect": "Allow",
            "Action": [
                "s3vectors:QueryVectors",
                "s3vectors:GetVectors",
                "s3vectors:PutVectors",
                "s3vectors:ListVectors"
            ],
            "Resource": [
                "arn:aws:s3vectors:aws-region:123456789012:bucket/amzn-s3-demo-vector-bucket/index/product-recommendations",
                "arn:aws:s3vectors:aws-region:123456789012:bucket/amzn-s3-demo-vector-bucket/index/content-similarity"
            ]
        },
        {
            "Sid": "AllowGetIndex",
            "Effect": "Allow",
            "Action": "s3vectors:GetIndex",
            "Resource": "arn:aws:s3vectors:aws-region:123456789012:bucket/amzn-s3-demo-vector-bucket/index/*"
        },
        {
            "Sid": "AllowIndexInspection",
            "Effect": "Allow",
            "Action": "s3vectors:ListIndexes",
```

```
            "Resource": "arn:aws:s3vectors:aws-region:123456789012:bucket/amzn-s3-demo-
vector-bucket"
        }
    ]
}
```

S3 Vectors resource-based policy examples

Resource-based policies are attached to a resource. You can create resource-based policies for vector buckets. Resource-based policies for S3 Vectors use the standard AWS policy format in JSON that you attach directly to vector buckets to control access to the bucket and its contents.

Unlike identity-based policies that are attached to users, groups, or roles, resource-based policies are attached to the resource itself (the vector bucket) and can grant permissions to principals from other AWS accounts. This makes them ideal for scenarios where you need to share vector data across organizational boundaries or implement fine-grained access controls based on the specific resource being accessed.

Resource-based policies are evaluated in combination with identity-based policies, and the effective permissions are determined by the union of all applicable policies. This means that a principal needs permission from both the identity-based policy (attached to their user/role) and the resource-based policy (attached to the bucket) to perform an action, unless the resource-based policy explicitly grants the permission.

Example 1: Cross-account access policy

This policy demonstrates how to grant specific permissions to users from different AWS accounts:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccountBucketAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/Admin"
            },
            "Action": [
                "s3vectors:CreateIndex",
                "s3vectors:ListIndexes",
                "s3vectors:QueryVectors",
```

```
                "s3vectors:PutVectors",
                "s3vectors:DeleteIndex"
            ],
            "Resource": [
                "arn:aws:s3vectors:aws-region:111122223333:bucket/amzn-s3-demo-vector-bucket/*",
                "arn:aws:s3vectors:aws-region:111122223333:bucket/amzn-s3-demo-vector-bucket"
            ]
        }
    ]
}
```

Example 2: Deny vector index level actions

This policy demonstrates how to deny specific vector index level actions to an IAM role:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyIndexLevelActions",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/External-Role-Name"
            },
            "Action": [
                "s3vectors:QueryVectors",
                "s3vectors:PutVectors",
                "s3vectors:DeleteIndex",
                "s3vectors:GetVectors",
                "s3vectors:GetIndex",
                "s3vectors:DeleteVectors",
                "s3vectors:CreateIndex",
                "s3vectors:ListVectors"
            ],
            "Resource": "arn:aws:s3vectors:aws-region:111122223333:bucket/amzn-s3-demo-vector-bucket/*"
        }
    ]
}
```

Example 3: Deny modification operations at both vector index and bucket levels

This policy demonstrates how to deny modification requests for both vector index and bucket-level actions by specifying multiple resources:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyModificationActionsAtBucketandIndexLevels",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/External-Role-Name"
            },
            "Action": [
                "s3vectors:CreateVectorBucket",
                "s3vectors:DeleteVectorBucket",
                "s3vectors:PutVectorBucketPolicy",
                "s3vectors:DeleteVectorBucketPolicy",
                "s3vectors:CreateIndex",
                "s3vectors:DeleteIndex",
                "s3vectors:PutVectors",
                "s3vectors:DeleteVectors"
            ],
            "Resource": [
                "arn:aws:s3vectors:aws-region:111122223333:bucket/amzn-s3-demo-vector-bucket/*",
                "arn:aws:s3vectors:aws-region:111122223333:bucket/amzn-s3-demo-vector-bucket"
            ]
        }
    ]
}
```

Data protection and encryption in S3 Vectors

Amazon S3 Vectors provides 99.999999999% (11 9s) of durability for your vector data, which ensures exceptional reliability for your vector storage needs. This durability is backed by the proven infrastructure of Amazon S3, which is designed to maintain data integrity and availability even in the face of hardware failures or other disruptions.

Data protection in S3 Vectors encompasses multiple layers of security controls designed to protect your vector data both at rest and in transit.

By default, all new vectors in Amazon S3 Vectors vector buckets use server-side encryption with Amazon S3 managed keys (SSE-S3). When you create a vector bucket with SSE-S3 encryption, all subsequent operations on the bucket automatically use encryption.

S3 Vectors also integrates with AWS Key Management Service (KMS) to provide flexible encryption key management options, allowing you to choose customer-managed keys for permission control and auditability.

When creating a vector index within a vector bucket, you can optionally override the vector bucket level encryption settings and provide an encryption configuration (SSE-S3 or KMS) at the vector index level. If no specific encryption is specified upon vector index creation, the index will inherit the encryption configuration from the vector bucket it belongs to.

Setting server-side encryption behavior for Amazon S3 vector buckets and indexes

Encryption configuration in S3 Vectors is a fundamental security setting to specify when you create a vector bucket. This design ensures that all vector data stored in the bucket is encrypted from the moment of creation. By default, the encryption configuration applies to all vectors, vector indexes, and metadata within the bucket, providing consistent protection across your entire vector dataset in a vector bucket. You can also optionally override the vector bucket level encryption settings and provide a dedicated encryption configuration (SSE-S3 or AWS KMS) at the vector index level.

> **Important**
>
> Encryption settings for a vector bucket can't be changed after the vector bucket is created. You must carefully consider your encryption requirements during the bucket creation process, including compliance requirements, key management preferences, and integration with existing security infrastructure.

When you set the SSE-S3 or SSE-KMS encryption type at the vector bucket level, by default it applies to all vector indexes and vectors within the bucket. The encryption configuration applies to not only the vector data itself but also all associated metadata.

You can also optionally override the vector bucket level encryption settings and provide a dedicated encryption configuration (SSE-S3 or KMS) at the vector index level. Encryption settings for a vector index can't be changed after the vector index is created.

Using SSE-S3 encryption

Server-side encryption with Amazon S3 managed keys (SSE-S3) provides a simple and effective encryption solution for vector buckets where AWS manages all aspects of the encryption process. This encryption method uses AES-256 encryption and is designed to provide strong security with minimal operational overhead, providing organizations with robust encryption without the complexity of the needs to manage encryption keys.

With SSE-S3, Amazon S3 handles the generation, rotation, and management of encryption keys automatically. SSE-S3 provides strong security with no additional configuration or ongoing management requirements. The encryption and decryption processes are handled automatically by the service, and there are no additional charges for using SSE-S3 encryption beyond the standard S3 Vectors pricing.

Using SSE-KMS encryption

Server-side encryption with AWS Key Management Service keys (SSE-KMS) provides enhanced control over encryption keys and enables detailed audit logging of key usage. This encryption method is ideal for organizations with strict compliance requirements, those that need to implement custom key rotation policies, or environments where detailed audit trails of data access are required.

SSE-KMS allows you to use customer managed keys (CMKs) for encrypting your vector data. Customer managed keys provide the highest level of control, allowing you to define key policies, enable or disable keys, and monitor key usage through AWS CloudTrail. This level of control makes SSE-KMS particularly suitable for regulated industries or organizations with specific data governance requirements.

When using SSE-KMS with customer managed keys, you have complete control over who can use the keys to encrypt and decrypt data. You can create detailed key policies that specify which users, roles, or services can access the keys.

Important considerations for SSE-KMS

- **KMS key format requirements:** S3 Vectors requires that you specify KMS keys using the full Amazon Resource Name (ARN) format. Key IDs or key aliases aren't supported.

- **Service principal permissions:** When you use customer managed keys with S3 Vectors, you must explicitly grant permissions to the S3 Vectors service principal to use your KMS key. This requirement ensures that the service can encrypt and decrypt your data on your behalf. The service principal that requires access is `indexing.s3vectors.amazonaws.com`.

Example: KMS key policy for S3 Vectors

To use a customer managed KMS key with S3 Vectors, you must update your key policy to include permissions for the S3 Vectors service principal. Here's a comprehensive key policy example.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowS3VectorsServicePrincipal",
            "Effect": "Allow",
            "Principal": {
                "Service": "indexing.s3vectors.amazonaws.com"
            },
            "Action": "kms:Decrypt",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3vectors:aws-region:123456789012:bucket/
*"
                },
                "StringEquals": {
                    "aws:SourceAccount": "123456789012"
                },
                "ForAnyValue:StringEquals": {
                    "kms:EncryptionContextKeys": ["aws:s3vectors:arn",
 "aws:s3vectors:resource-id"]
                }
            }
        },
        {
            "Sid": "AllowApplicationAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::123456789012:role/VectorApplicationRole",
```

```
                    "arn:aws:iam::123456789012:user/DataScientist"
                ]
            },
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3vectors.aws-region.amazonaws.com"
                },
                "ForAnyValue:StringEquals": {
                    "kms:EncryptionContextKeys": ["aws:s3vectors:arn",
 "aws:s3vectors:resource-id"]
                }
            }
        }
    ]
}
```

- **Required KMS permissions:**
  - S3 Vectors service principal permission:
    - `kms:Decrypt` – Required by the S3 Vectors service principal (`indexing.s3vectors.amazonaws.com`) on your customer managed key to maintain and optimize the index in background operations
  - IAM principal permissions:
    - `kms:Decrypt` – Required for all vector-level operations (PutVectors, GetVectors, QueryVectors, DeleteVectors, ListVectors)
    - `kms:GenerateDataKey` – Required to create a vector bucket by using the customer managed key
- **Cross-account access considerations:** When implementing cross-account access patterns with SSE-KMS, you must ensure that the KMS key policy allows access from the appropriate principals in other accounts. The key ARN format becomes particularly important in cross-account scenarios, as it provides an unambiguous reference to the key regardless of the account context from which it's being accessed.

Setting encryption in S3 Vectors

This topic explains how to set the encryption configuration for your S3 vector buckets and indexes.

Before you begin, make sure you have the following:

- Appropriate permissions to view bucket and index properties.

Using the S3 console

To configure encryption for a vector bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Vector buckets**.
3. Choose **Create vector bucket**.
4. For **Bucket name**, enter a name for your bucket.

   The bucket name must:

   - Be unique within your account for this AWS Region
   - Be between 3 and 63 characters long
   - Consist only of lowercase letters, numbers, and hyphens (-)
5. For **Encryption**, choose

   - **Specify encryption type** – Choose a specific encryption method:
     - **Server-side encryption with Amazon S3 managed keys (SSE-S3)** – With SSE-S3, Amazon S3 handles the generation, rotation, and management of encryption keys automatically.
     - **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** – Similar to SSE-S3, but uses customer managed keys (CMKs) in AWS KMS, giving you more control over your keys. For more information about customer managed keys, see Customer managed keys in the *AWS Key Management Service Developer Guide*.

       If you select this option, under **AWS KMS key**, choose one of the following options:

       - **Choose from your AWS KMS keys** – Select an existing KMS key from the dropdown list
       - **Enter AWS KMS key ARN** – Enter the Amazon Resource Name (ARN) of a KMS key

- **Create a KMS key** – Create a new customer managed key in the AWS KMS console. For more information, see Creating symmetric customer managed keys in the *AWS Key Management Service Developer Guide*.

> **Note**
>
> The following requirements apply to the KMS key:
>
> - AWS KMS key ID must not be empty
> - Your KMS key must be in the same Region where this bucket is being created
> - AWS KMS key ARN must start with "arn:aws:kms:"

> **Important**
>
> Encryption settings can't be changed after the vector bucket is created.

6. If you chose **Enter AWS KMS key ARN**, enter the ARN in the text field provided.
7. If you chose **Create a KMS key**, the console opens the AWS KMS console in a new tab. For instructions on creating a KMS key, see Creating symmetric customer managed keys in the *AWS Key Management Service Developer Guide*.
8. Choose **Create vector bucket**.

> **Important**
>
> When using KMS encryption, ensure that the IAM principals that need to access objects in the bucket have the necessary KMS permission (kms:Decrypt) for the selected KMS key.

To configure encryption for a vector index

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Vector buckets**.
3. In the list of vector buckets, choose the name of the bucket where you want to create a vector index.

4. Choose **Create vector index**.
5. For **Vector index name**, enter a name for your vector index.

   Vector index names must be unique within the vector bucket. Index name must be between 3 and 63 characters. Valid characters are lowercase letters (a-z), numbers (0-9), hyphens (-), and dots (.). For more information about the vector index naming requirements, see Vector bucket naming rules.
6. For **Dimension**, enter the number of values in each vector.

   > **Note**
   >
   > - The value for **Dimension** determines how many numerical values each vector will contain.
   > - All vectors added to this index must have exactly this number of values.
   > - Dimension must be between 1 and 4096.
   > - A larger dimension requires more storage space.
   > - Choose based on your embedding model's output dimensions.
   >
   > For more information about the dimension requirements, see Limitations and restrictions.
7. For **Distance metric**, choose one of the following options:

   - **Cosine** – Measures the cosine of the angle between vectors. Best for normalized vectors and when direction matters more than magnitude
   - **Euclidean** – Measures the straight-line distance between vectors. Best when both direction and magnitude are important.
8. (Optional) Under **Non-filterable metadata**, configure metadata keys that will be stored but not used for filtering:

   To add non-filterable metadata keys:

   a. Choose **Add key**.

   b. Enter a key name (1-63 characters and unique within this vector index).

   c. Repeat to add additional keys (maximum 10 keys).

> **Note**
>
> You can attach filterable metadata as key-value pairs to each vector when you insert vector data after you create a vector index. By default, all metadata keys that are attached to vectors are filterable and can be used as filters in a similarity query. Only metadata keys that are specified as non-filterable during vector index creation are excluded from filtering. For more information about metadata size limits per vector, including both total and filterable metadata constraints, see Limitations and restrictions.

9. For **Encryption**, choose **Specify encryption type** and then choose one of the following options:

   - **Use bucket settings for encryption** – Amazon S3 applies the vector bucket encryption settings to encrypt vector data in the vector index.
   - **Override bucket settings for encryption** – Specify a specific encryption type for the vector index:
     - **Server-side encryption with Amazon S3 managed keys (SSE-S3)** – With SSE-S3, Amazon S3 handles the generation, rotation, and management of encryption keys automatically.
     - **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** – Similar to SSE-S3, but uses customer managed keys (CMKs) in AWS KMS, giving you more control over your keys. For more information about customer managed keys, see Customer managed keys in the *AWS Key Management Service Developer Guide*.

       If you select this option, under **AWS KMS key**, choose one of the following options:

       - **Choose from your AWS KMS keys** – Select an existing KMS key from the dropdown list
       - **Enter AWS KMS key ARN** – Enter the Amazon Resource Name (ARN) of a KMS key
       - **Create a KMS key** – Create a new customer managed key in the AWS KMS console. For more information, see Creating symmetric customer managed keys in the *AWS Key Management Service Developer Guide*.

       > **Note**
       >
       > The following requirements apply to the KMS key:
       >
       > - AWS KMS key ID must not be empty.

- Your KMS key must be in the same Region where this bucket is being created.
- AWS KMS key ARN must start with "arn:aws:kms:"

Important

- Encryption settings can't be changed after the vector index is created.
- If you chose **Enter AWS KMS key ARN**, enter the ARN in the text field provided.
- If you chose **Create a KMS key**, the console opens the AWS KMS console in a new tab. For instructions on creating a KMS key, see *Creating symmetric customer managed keys* in the *AWS Key Management Service Developer Guide*.

Important

When using KMS encryption, ensure that the IAM principals that need to access objects in the bucket have the necessary KMS permission (kms:Decrypt) for the selected KMS key.

10. Under **Tags (Optional)**, you can add tags as key-value pairs to help track and organize vector index costs using AWS Billing and Cost Management. Enter a **Key** and a **Value**. To add another tag, choose **Add Tag**. You can enter up to 50 tags for a vector index. For more information, see Using tags with S3 vector buckets.
11. Review your configuration carefully.

Note

These settings can't be changed after creation.

12. Choose **Create vector index**.

Using the AWS CLI

The following example shows how to create a vector bucket with the SSE-S3 encryption configuration by using the AWS CLI. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3vectors create-vector-bucket \
        --vector-bucket-name "amzn-s3-demo-vector-bucket" \
        --encryption-configuration '{"sseType": "AES256"}'
```

The following examples shows how to create a vector bucket that uses the SSE-KMS encryption configuration with a customer managed key. To use this example, replace the *user input placeholders* with your own information.

```
aws s3vectors create-vector-bucket \
        --vector-bucket-name "amzn-s3-demo-vector-bucket" \
        --encryption-configuration '{"sseType": "aws:kms", "kmsKeyArn":
 "arn:aws:kms:us-east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"}'
```

The following example shows how to create a vector index with the SSE-S3 encryption configuration by using the AWS CLI. To use this example, replace the *user input placeholders* with your own information.

```
aws s3vectors create-index \
        --vector-bucket-name "amzn-s3-demo-vector-bucket" \
        --index-name "amzn-s3-demo-vector-index" \
        --encryption-configuration '{"sseType": "AES256"}'
```

The following examples shows how to create a vector index that uses the SSE-KMS encryption configuration with a customer managed key. To use this example, replace the *user input placeholders* with your own information.

```
aws s3vectors create-index \
        --vector-bucket-name "amzn-s3-demo-vector-bucket" \
        --index-name "amzn-s3-demo-vector-index" \
        --encryption-configuration '{"sseType": "aws:kms",
 "kmsKeyArn": "arn:aws:kms:us-
east-1:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890abc"}'
```

The following examples shows will create a vector index using the encryption settings of the vector bucket. To use this example, replace the *user input placeholders* with your own information.

```
aws s3vectors create-index \
        --vector-bucket-name "amzn-s3-demo-vector-bucket" \
        --index-name "amzn-s3-demo-vector-index" \
```

Viewing encryption configuration in S3 Vectors

After creating your vector bucket, you can verify the encryption configuration using the console. Alternatively, you can use the GetVectorBucket and GetIndex API operations through the AWS REST API, AWS CLI, or AWS SDKs.

Using the AWS CLI

Use the `get-vector-bucket` command to retrieve detailed bucket information, including encryption configuration. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3vectors get-vector-bucket \
  --vector-bucket-name amzn-s3-demo-vector-bucket
```

Use the `get-index` command to retrieve detailed vector index information, including encryption configuration. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3vectors get-index \
  --vector-bucket-name amzn-s3-demo-vector-bucket
  --index-name amzn-s3-demo-vector-index
```

VPC endpoints for S3 Vectors

To access S3 Vectors from your virtual private cloud (VPC), Amazon S3 supports interface VPC endpoints by using AWS PrivateLink (PrivateLink). PrivateLink provides private connectivity between your VPC and S3 Vectors without requiring an internet gateway or NAT device. Interface endpoints are represented by one ore more elastic network interfaces (ENIs) that are assigned private IP addresses from subnets in your VPC. Requests to S3 Vectors over interface endpoints stay on the AWS network.

You can also access interface endpoints in your VPC from on-premises applications through AWS Direct Connect or AWS Virtual Private Network (AWS VPN). For more information about how to connect your VPC with your on-premises network, see the *AWS Direct Connect User Guide* and the *AWS Site-to-Site VPN User Guide*. For general information about interface endpoints, see Access an AWS service using an interface VPC endpoint in the *AWS PrivateLink Guide*.

Benefits of using PrivateLink with S3 Vectors

Using PrivateLink with S3 Vectors provides several security and operational benefits:

- **Enhanced security**: Traffic between your VPC and S3 Vectors remains within the AWS network and doesn't traverse the internet.
- **Simplified network architecture**: Access S3 Vectors without configuring internet gateways, NAT devices, or VPN connections.
- **Granular access control**: Use VPC endpoint policies to control which vector buckets and vector indexes can be accessed through the endpoint.
- **Compliance support**: Meet regulatory requirements that mandate private network connectivity for sensitive data.

VPC endpoint DNS names and resolution

When you create a VPC endpoint, S3 Vectors generates two types of endpoint-specific DNS names: Regional and Zonal.

The Regional and Zonal DNS names of interface VPC endpoints for S3 Vectors are as follows:

- **Regional DNS name**: `vpce-`*`1a2b3c4d-5e6f`*`.s3vectors.`*`region`*`.vpce.amazonaws.com` - The regional VPC endpoint DNS name. Always resolve to private IP addresses.
- **Zonal DNS name**: `vpce-`*`1a2b3c4d-5e6f`*`-`*`availability_zone_code`*`.s3vectors.`*`region`*`.vpce.amazonaws.com` - Zone-specific VPC endpoint DNS names. Always resolve to private IP addresses.

You can also use the DNS name of the public endpoint `s3vectors.`*`region`*`.api.aws` as the private DNS name of the endpoint service if you have private DNS enabled for the VPC endpoint.

IP addressing for interface endpoints

S3 Vectors regional, zonal, and private DNS endpoints support IPv4, IPv6, and dualstack IP types for AWS PrivateLink. For more information, see IP address types and DNS record IP type for AWS services in the *AWS PrivateLink Guide*.

The following are some things you should know before trying to access S3 Vectors vector indexes and vector buckets over IPv6 in your VPC:

- The client you use to access vectors and your S3 Vectors client must both have dual-stack enabled.
- If your VPC security group doesn't have IPv6 set up, you'll need to configure a rule to allow IPv6 traffic. For more information, see Step 3: Update your security group rules in the *VPC User Guide* and Configure security group rules in the *VPC User Guide*.
- If your VPC doesn't have IPv6 CIDRs assigned, you will need to manually add an IPv6 CIDR block to your VPC. For more information, see Add IPv6 support for your VPC in the *AWS PrivateLink Guide*.
- If you use IP address filtering IAM policies, they must be updated to handle IPv6 addresses. For more information about managing access permissions with IAM, see Identity and Access management in S3 Vectors.

Creating a VPC interface endpoint for S3 Vectors

You can create a VPC interface endpoint for S3 Vectors using the VPC console, AWS CLI, AWS SDKs, or AWS API.

Using the S3 console

1. Open the VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation pane, choose **Endpoints**.
3. Choose **Create endpoint**.
4. For **Service category**, choose **AWS services**.
5. For **Services**, search for `s3vectors` and select `com.amazonaws.region.s3vectors`.
6. For **VPC**, select the VPC where you want to create the endpoint.
7. (Optional) Under **Additional settings**, for **Enable DNS name**, choose whether to enable the private DNS feature. When enabled, requests that use the public service endpoint (`s3vectors.region.api.aws`), such as requests made through AWS SDKs, resolve to your VPC endpoint instead of the public endpoint.
8. For **Subnets**, select the subnets where you want to create the endpoint network interfaces.
9. For **IP address type**, choose the IP address type for the endpoint:
   - **IPv4**: Assign IPv4 addresses to the endpoint network interfaces. This option is supported only if all selected subnets have IPv4 address ranges.

- **IPv6**: Assign IPv6 addresses to the endpoint network interfaces. This option is supported only if all selected subnets are IPv6 only subnets.
- **Dualstack**: Assign both IPv4 and IPv6 addresses to the endpoint network interfaces. This option is supported only if all selected subnets have both IPv4 and IPv6 address ranges.

10. For **Security groups**, select the security groups to associate with the endpoint network interfaces.
11. (Optional) For **Policy**, you can attach a VPC endpoint policy to control access to S3 Vectors through the endpoint. To allow all operations by all principals on all S3 Vectors resources over the interface endpoint, choose **Full access**. To restrict access, choose **Custom** and enter a policy. For more information, see Control access to VPC endpoints using endpoint policies in the AWS PrivateLink Guide. If you don't attach a policy, the default policy allows full access.
12. Choose **Create endpoint**.

Using the AWS CLI

To create a new VPC endpoint that returns both IPv4 and IPv6 for S3 Vectors, use the following example CLI command. For more information, see create-vpc-endpoint.

```
aws ec2 create-vpc-endpoint \
    --vpc-id vpc-12345678 \
    --service-name com.amazonaws.region.s3vectors \
    --vpc-endpoint-type Interface \
    --subnet-ids subnet-12345678 subnet-87654321 \
    --security-group-ids sg-12345678 \
    --ip-address-type dualstack \
    --private-dns-enabled
```

The `--private-dns-enabled` parameter enables the private DNS feature. When enabled, requests to `s3vectors.region.api.aws` will route through your VPC endpoint.

For more information about creating VPC endpoints, see Create a VPC endpoint in the *VPC User Guide*.

VPC endpoint policies for S3 Vectors

Similar to resource-based policies, you can attach an endpoint policy to your VPC endpoint to control access to vector indexes and vector buckets. For more information about the endpoint policies, see Control access to VPC endpoints using endpoint policies in the AWS PrivateLink Guide.

Example VPC endpoint policies

The following example VPC endpoint policy allows access to all S3 Vectors operations for all principals:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3vectors:*"
      ],
      "Resource": "*"
    }
  ]
}
```

The following example VPC endpoint policy restricts access to a specific vector bucket:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3vectors:GetVectorBucket",
        "s3vectors:ListIndexes",
        "s3vectors:GetIndex",
        "s3vectors:QueryVectors",
        "s3vectors:GetVectors"
      ],
      "Resource": [
        "arn:aws:s3vectors:us-west-2:111122223333:bucket/amzn-s3-demo-vector-bucket",
        "arn:aws:s3vectors:us-west-2:111122223333:bucket/amzn-s3-demo-vector-bucket/*"
      ]
    }
  ]
}
```

The following example VPC endpoint policy allows access only during business hours using the `aws:CurrentTime` condition key:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3vectors:*",
      "Resource": "*",
      "Condition": {
        "DateGreaterThan": {
          "aws:CurrentTime": "08:00Z"
        },
        "DateLessThan": {
          "aws:CurrentTime": "18:00Z"
        }
      }
    }
  ]
}
```

Configuring S3 Vectors clients for VPC endpoints

When using VPC endpoints with S3 Vectors, you can configure your S3 Vectors clients to use either the service DNS name or the VPC endpoint DNS name.

Using the AWS SDKs

SDK for Python (Boto3)

The following example shows how to configure the S3 Vectors client in SDK for Python (Boto3) to use a VPC endpoint:

```
import boto3

# Using service DNS name (requires private DNS feature enabled on VPC endpoint)
s3vectors_client = boto3.client(
    's3vectors',
    region_name='us-west-2',
    endpoint_url='https://s3vectors.us-west-2.api.aws'
)
```

```
# Using VPC endpoint DNS name
s3vectors_client = boto3.client(
    's3vectors',
    region_name='us-west-2',
    endpoint_url='https://vpce-12345678.s3vectors.us-west-2.vpce.amazonaws.com'
)
```

Troubleshooting VPC endpoints

If you're experiencing issues with your interface VPC endpoint, consider the following troubleshooting steps:

- **DNS resolution**: Verify that DNS queries for the endpoint resolve to private IP addresses within your VPC CIDR range when using private DNS.
- **Security groups**: Ensure that the security group associated with the VPC endpoint allows inbound HTTPS traffic (port 443) from your VPC resources.
- **Route tables**: Verify that your subnet route tables don't have conflicting routes that might redirect traffic away from the VPC endpoint.
- **VPC endpoint policy**: Check that your VPC endpoint policy allows the necessary S3 Vectors actions and resources.
- **Client configuration**: If the private DNS feature is disabled, configure your S3 Vectors client to use the VPC endpoint DNS name instead of the service DNS name.

Monitoring VPC endpoint usage

You can monitor your S3 Vectors VPC endpoint usage through CloudTrail NetworkActivity events logs.

For more information about S3 Vectors logging, see Logging with AWS CloudTrail for S3 Vectors.

Logging with AWS CloudTrail for S3 Vectors

Amazon S3 Vectors is integrated with AWS CloudTrail, a service that provides a record of actions that are taken by a user, role, or an AWS service. CloudTrail captures all API calls for S3 Vectors as events. Using the information that's collected by CloudTrail, you can determine the request that was made to S3 Vectors, the IP address from which the request was made, when it was made, and

additional details. When a supported event activity occurs in S3 Vectors, that activity is recorded in a CloudTrail event. You can use CloudTrail trail to log management events and data events for S3 Vectors.

To learn more about CloudTrail, see the CloudTrail User Guide.

S3 Vectors information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in S3 Vectors, that activity is recorded in a CloudTrail event along with other AWS service events in Event history. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for S3 Vectors, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see Overview for Creating a Trail, Configuring Amazon SNS Notifications for CloudTrail, Receiving CloudTrail Log Files from Multiple Regions, and Receiving CloudTrail Log Files from Multiple Accounts in the *CloudTrail User Guide*.

All S3 Vectors API actions are logged by CloudTrail and are documented in the Amazon S3 Vectors API Reference. For example, calls to the CreateVectorBucket, CreateIndex, and QueryVectors actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element in the *CloudTrail User Guide*.

CloudTrail management events for S3 Vectors

Management events provide information about management operations that are performed on resources in your AWS account. These are also known as control plane operations. By default, CloudTrail logs management events.

For S3 Vectors, CloudTrail logs the following management events:

- CreateVectorBucket
- DeleteVectorBucket
- GetVectorBucket
- ListVectorBuckets
- PutVectorBucketPolicy
- GetVectorBucketPolicy
- DeleteVectorBucketPolicy
- CreateIndex
- DeleteIndex
- GetIndex
- ListIndexes

The `eventSource` for S3 Vectors management events and data events is `s3vectors.amazonaws.com`.

For more information about management events, see Logging management events in the *CloudTrail User Guide*.

CloudTrail data events for S3 Vectors

Data events provide information about the resource operations performed on or in a resource. These are also known as data plane operations. By default, CloudTrail doesn't log data events. However, you can configure trails to log data events for S3 Vectors resources.

When you configure your trail to log data events, you can specify the S3 Vectors resource type. S3 Vectors supports the following resource types for data events:

- `AWS::S3Vectors::VectorBucket` - Logs data events for all vector indexes in the specified vector buckets

- `AWS::S3Vectors::Index` - Logs data events for specific vector indexes

For S3 Vectors, CloudTrail logs the following data events:

Vector data operations:

- PutVectors - Logs when vectors are added to a vector index
- GetVectors - Logs when vectors are retrieved from a vector index
- DeleteVectors - Logs when vectors are deleted from a vector index
- ListVectors - Logs when vectors in a vector index are listed
- QueryVectors - Logs when similarity queries are performed on a vector index

The `eventSource` for S3 Vectors data events is `s3vectors.amazonaws.com`.

Enabling data event logging for S3 Vectors

You can enable data event logging for S3 Vectors resources when you create or update a CloudTrail trail. You can specify logging for all vector buckets and vector indexes in your account, or you can specify individual vector buckets or vector indexes. For detailed steps about creating a trail, see Creating a trail in the *CloudTrail User Guide*.

To enable data event logging for all S3 Vectors resources:

- When creating or updating a trail, choose **Data events**.
- For **Resource type**, choose `AWS::S3Vectors::VectorBucket`.
- For **Resource ARN**, enter `arn:aws:s3vectors:_:_:bucket/*` to log events for all vector buckets, or specify individual vector bucket ARNs (for example, `arn:aws:s3vectors:`*`us-east-1`*`:`*`123456789012`*`:bucket/`*`amzn-s3-demo-vector-bucket`*).

To enable data event logging for specific vector indexes:

- When creating or updating a trail, choose **Data events**.
- For **Resource type**, choose `AWS::S3Vectors::Index`.
- For **Resource ARN**, enter the ARN of the specific vector index, such as: `arn:aws:s3vectors:`*`us-east-1`*`:`*`123456789012`*`:bucket/`*`amzn-s3-demo-vector-bucket`*`/index/`*`my-index`*.

For more information about data events, see Logging data events in the *CloudTrail User Guide*.

Topics

- CloudTrail log file example for S3 Vectors

CloudTrail log file example for S3 Vectors

The following examples show CloudTrail log entries for S3 Vectors data events. Data events are logged when you perform operations on vector data within your vector indexes.

Example: CloudTrail log file for `GetVectors` data event


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "123456789012",
        "arn": "arn:aws:iam::123456789012:user/myUserName",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "myUserName"
    },
    "eventTime": "2024-11-22T17:12:25Z",
    "eventSource": "s3vectors.amazonaws.com",
    "eventName": "GetVectors",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "[aws-cli/2.18.5]",
    "requestParameters": {
        "vectorBucketName": "amzn-s3-demo-vector-bucket",
        "returnMetadata": "false",
        "indexName": "111aa1111-22bb-33cc-44dd-5555eee66ffff",
        "returnData": "false"
    },
    "responseElements": null,
    "additionalEventData": {
        "SignatureVersion": "SigV4",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256"
    },
    "requestID": "07D681123BD12AED",
    "eventID": "f2b287f3-0df1-1234-a2f4-c4bdfed47657",
    "readOnly": true,
```

```
    "resources": [{
        "accountId": "123456789012",
        "type": "AWS::S3Vectors::VectorBucket",
        "ARN": "arn:aws:s3vectors:us-east-1:123456789012:bucket/amzn-s3-demo-vector-
bucket"
    }, {
        "accountId": "123456789012",
        "type": "AWS::S3Vectors::Index",
        "ARN": "arn:aws:s3vectors:us-east-1:123456789012:bucket/amzn-s3-demo-vector-
bucket/index/111aa1111-22bb-33cc-44dd-5555eee66ffff"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "444455556666",
    "eventCategory": "Data",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "client-host"
    }
}
```

Understanding S3 Vectors log file entries

CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

S3 Vectors CloudTrail log entries contain the following key elements:

- **eventSource** - Always `s3vectors.amazonaws.com` for S3 Vectors events.
- **eventName** - The S3 Vectors API operation that was performed.
- **eventCategory** - Either `Management` for control plane operations or `Data` for data plane operations.
- **readOnly**: `true` for read operations (for examples, GetVectors, QueryVectors, ListVectors) and `false` for write operations (PutVectors, DeleteVectors).
- **resources** - The S3 Vectors resources involved in the operation, including vector buckets and vector indexes.
- **requestParameters** - The parameters that were sent with the request.

- **responseElements** - The response elements returned by the S3 Vectors service.

Working with Amazon S3 Files

Topics

- What is S3 Files?
- How does S3 Files work?
- Are you a first-time user of S3 Files?
- Key concepts
- Features
- How are you billed for S3 Files?
- Prerequisites for S3 Files
- Tutorial: Getting started with S3 Files
- Mounting your S3 buckets on compute resources
- Creating and managing S3 Files resources
- Understanding how synchronization works
- Monitoring and auditing S3 Files
- Performance specifications
- Security for S3 Files
- How S3 Files is metered
- S3 Files best practices
- Unsupported features, limits, and quotas
- Troubleshooting S3 Files

What is S3 Files?

S3 Files is a shared file system that connects any AWS compute resource directly with your data in Amazon S3. It provides fast, direct access to all of your S3 data as files with full file system semantics and low-latency performance, without your data ever leaving S3. Every file-based application, agent, and team can access and work with your S3 data as a file system using the tools they already depend on. Built using Amazon EFS, S3 Files gives you the performance and simplicity of a file system with the scalability, durability, and cost-effectiveness of S3. You can read, write,

and organize data using file and directory operations, while S3 Files manages the synchronization of changes between your bucket and file system.

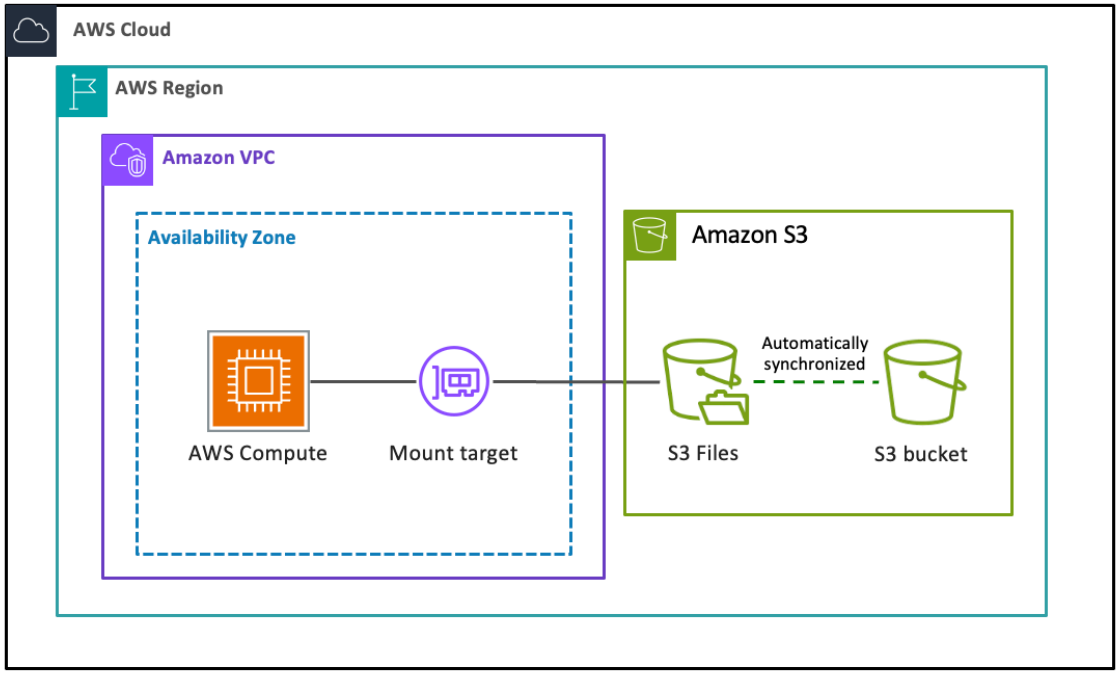
How does S3 Files work?

When you create an S3 file system linked to your S3 bucket or to a prefix within it and mount it on a compute resource such as an EC2 instance or a Lambda function, S3 Files first presents a traversable view of your bucket's objects as files. As you navigate through directories and open files, associated metadata and contents are placed onto the file system's high-performance storage. When you read files, S3 Files loads file contents onto the high-performance storage on demand without duplicating your entire dataset. When you write data, your writes go to the high-performance storage and are synchronized back to your S3 bucket. S3 Files intelligently translates your file system operations into efficient S3 requests on your behalf. Many read operations bypass the file system entirely, with data served directly from S3.

You can configure the file size threshold for what gets loaded onto the high-performance storage (default <128 KiB), as latencies matter most for small files. S3 Files streams file reads directly from your S3 bucket in two cases: when the file's data is not stored in the file system's high-performance storage, and for large reads >= 1 MiB, even when the data also resides on the file system's high-performance storage. The S3 bucket is optimized for high throughput while the file system's high-performance storage layer is optimized for low-latency access. S3 Files asynchronously imports data for small files (< 128 KiB by default) to the file system's high-performance storage for low latency access on subsequent reads. Recently modified data that has not yet been synchronized to S3 is always served from the file system. For more information, see Customizing synchronization for S3 Files.

Data that has not been read within a configurable window (1 to 365 days, default 30) automatically expires from the high-performance storage. Your authoritative data always remains in S3, and background synchronization keeps the file system and bucket consistent in both directions. For more information, see Understanding how synchronization works.

Supported compute services to mount your S3 file systems are Amazon EC2, AWS Lambda, Amazon EKS, and Amazon ECS. For more information, see Mounting your S3 buckets on compute resources.



Are you a first-time user of S3 Files?

If you are a first-time user of S3 Files, create your first S3 file system using the S3 Console or the AWS CLI by following the Tutorial: Getting started with S3 Files.

Key concepts

The following terms are used throughout S3 Files documentation:

File system

A shared file system linked to your S3 bucket.

High-performance storage

The low-latency storage layer within your file system where actively used file data and metadata reside. S3 Files automatically manages this storage, copying data onto it when you access files and removing data that has not been read within a configurable expiration window. You pay a storage rate for data residing on the high-performance storage.

Synchronization

The process by which S3 Files keeps your active working dataset and your changes consistent between your file system and S3 bucket. Importing copies data from your S3 bucket onto the

file system. Exporting copies changes you make through the file system back to your S3 bucket. S3 Files performs synchronization automatically in both directions.

Mount target

A mount target provides network access to your file system within a single Availability Zone in your VPC. You need at least one mount target to access your file system from compute resources, and you can create a maximum of one mount target per Availability Zone.

Access point

Access points are application-specific entry points to a file system that simplify managing data access at scale for shared datasets. You can use access points to enforce user identities and permissions for all file system requests that are made through the access point. When you create a file system using the AWS Management Console, S3 Files automatically creates one access point for the file system.

Features

High performance without full data replication

S3 Files delivers low-latency file access by copying only your active working set onto the file system's high performance storage, not your entire dataset. Small, frequently accessed files are served from the high-performance storage at sub-millisecond to single-digit millisecond latencies. Large reads are streamed directly from S3 at up to terabytes per second of aggregate throughput. This means you get file system performance for interactive workloads and S3 throughput for streaming workloads, without paying to store or import data you are not using or doesn't benefit from low latency. For more information, see Performance specifications.

Intelligent read routing

S3 Files automatically routes read requests to the storage layer (S3 file system or S3 bucket) best suited for them, while maintaining full file system semantics including consistency, locking, and POSIX permissions. Small, random reads of actively used files are served from the high-performance storage for low latency. Large sequential reads and reads of data not on the file system are served directly from your S3 bucket for high throughput, with no file system data charge.

Automatic synchronization

S3 Files automatically keeps your file system and S3 bucket consistent in both directions. Changes you make through the file system are copied back to your S3 bucket, and changes

made directly to your S3 bucket are reflected in your file system's view. You can customize synchronization behavior, including what data is imported and how long it stays on the file system. For more information, see Understanding how synchronization works.

Scalable performance

S3 Files automatically scales throughput and IOPS to match your workload activity. You do not need to provision or manage performance capacity and you pay only for what you use.

Regional durability

Data written to the high performance storage layer has the same durability as Amazon S3. It stores data redundantly across multiple geographically separated Availability Zones within the same AWS Region, providing high durability and availability for your data.

Encryption

S3 Files encrypts all data in transit using TLS and all data at rest using AWS KMS keys. You can use AWS owned keys (default) or your own customer managed keys. For more information, see Encryption.

File system semantics

S3 Files supports the NFS version 4.2 and 4.1 protocols. It provides file-system-access semantics, such as read-after-write data consistency, file locking, and POSIX permissions.

How are you billed for S3 Files?

You pay a storage rate for the fraction of active data resident on the high-performance storage, and you pay file system access charges for reading from and writing to your file system's high performance storage. S3 Files streams file reads directly from your S3 bucket in two cases: when the file's data is not stored in the file system's high-performance storage, and for large reads >= 1 MiB, even when the data also resides on the file system's high-performance storage. The S3 bucket is optimized for high throughput while the file system's high-performance storage layer is optimized for low-latency access. S3 Files asynchronously imports data for small files (< 128 KiB by default) to the file system's high-performance storage for low latency access on subsequent reads. These reads incur only standard S3 GET request cost with no file system access charge. The file system access charges apply to synchronization operations: importing data onto the file system incurs write charges, and exporting changes back to S3 incurs read charges. For more information, see How S3 Files is metered. For current pricing, see the S3 Files pricing page.

Prerequisites for S3 Files

Before you begin using S3 Files, make sure that you have completed the following prerequisites.

AWS account and compute setup

- You have an AWS account.
- You have a compute resource and an S3 general purpose bucket in your desired AWS Region where you want to create your file system. For more information, see Creating a general purpose bucket.
- Your S3 bucket has versioning enabled. S3 Files requires S3 Versioning to synchronize changes between your file system and your S3 bucket. For more information, see Enabling versioning on buckets.
- Your S3 bucket must use one of the following encryption types: Server-side encryption with Amazon S3 managed keys (SSE-S3) or Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS).

S3 Files client

To use S3 Files with Amazon EC2, you must install the client `amazon-efs-utils`, a shared open-source collection of tools for Amazon EFS and Amazon S3 Files. To work with S3 Files, you need `amazon-efs-utils` version 3.0.0 or above. The client includes a mount helper program that simplifies mounting S3 file systems and enables Amazon CloudWatch metrics for monitoring your file system's mount status.

Step 1: Install the client

- Access the terminal for your Amazon EC2 instance through Secure Shell (SSH), and log in with the appropriate user name. For more information, see Connect to your EC2 instance in the *Amazon Elastic Compute Cloud User Guide*.
- For those using Amazon Linux, do the following to install efs-utils from Amazon's repositories:

```
sudo yum -y install amazon-efs-utils
```

- If you use other supported Linux distributions, you can do the following:

```
curl https://amazon-efs-utils.aws.com/efs-utils-installer.sh | sudo sh -s -- --
install
```

- For other Linux distributions, see On other Linux distributions in the amazon-efs-utils README on GitHub.

Step 2: Install botocore

The `amazon-efs-utils` client uses botocore to interact with other AWS services. For example, you need to install botocore to use Amazon CloudWatch to monitor your file system. For instructions on installing and upgrading botocore, see Installing botocore in the amazon-efs-utils README on GitHub.

Enabling FIPS mode for S3 Files

If you need to comply with Federal Information Processing Standards (FIPS), then you must enable FIPS mode in the client. Enabling the FIPS mode involves modifying the `s3files-utils.conf` file on the operating system.

Follow these steps to enable FIPS mode in the client for S3 Files:

1. Using your text editor of choice, open the `/etc/amazon/efs/s3files-utils.conf` file.
2. Find the line containing the following text:

```
fips_mode_enabled = false
```

3. Change the text to the following:

```
fips_mode_enabled = true
```

4. Save your changes.

IAM roles and policies

To use S3 Files, you must configure IAM roles and attached policies for two purposes:

- Accessing your bucket from the file system
- Attaching your file system to AWS compute resources

IAM role for accessing your bucket from the file system

When you create an S3 Files file system, you specify an IAM role that S3 Files assumes to read from and write to your S3 bucket. This role grants S3 Files permission to synchronize changes between your file system and your bucket, including Amazon EventBridge rules that trigger synchronization. You must also make sure that your S3 bucket policy does not deny access to your compute resources.

S3 Files requires S3 Versioning on your bucket. It uses version-specific API operations such as `s3:GetObjectVersion` and `s3:GetObjectVersionTagging` in addition to `s3:GetObject`. The following policy uses wildcards (for example, `s3:GetObject*` rather than `s3:GetObject`) to cover both versioned and non-versioned actions.

> **Note**
>
> When you create a file system using the AWS Management Console, S3 Files automatically creates this IAM role with the required permissions.

This IAM role requires the following:

- An inline policy as follows:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3BucketPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:ListBucketVersions"
            ],
            "Resource": "arn:aws:s3:::bucket",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "accountId"
                }
            }
        },
        {
```

```
            "Sid": "S3ObjectPermissions",
            "Effect": "Allow",
            "Action": [
                "s3:AbortMultipartUpload",
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:List*",
                "s3:PutObject*"
            ],
            "Resource": "arn:aws:s3:::bucket/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "accountId"
                }
            }
        },
        {
            "Sid": "UseKmsKeyWithS3Files",
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncryptFrom",
                "kms:ReEncryptTo"
            ],
            "Condition": {
                "StringLike": {
                    "kms:ViaService": "s3.region.amazonaws.com",
                    "kms:EncryptionContext:aws:s3:arn": [
                        "arn:aws:s3:::bucket",
                        "arn:aws:s3:::bucket/*"
                    ]
                }
            },
            "Resource": "arn:aws:kms:region:accountId:*"
        },
        {
            "Sid": "EventBridgeManage",
            "Effect": "Allow",
            "Action": [
                "events:DeleteRule",
                "events:DisableRule",
                "events:EnableRule",
```

```
                "events:PutRule",
                "events:PutTargets",
                "events:RemoveTargets"
            ],
            "Condition": {
                "StringEquals": {
                    "events:ManagedBy": "elasticfilesystem.amazonaws.com"
                }
            },
            "Resource": [
                "arn:aws:events:*:*:rule/DO-NOT-DELETE-S3-Files*"
            ]
        },
        {
            "Sid": "EventBridgeRead",
            "Effect": "Allow",
            "Action": [
                "events:DescribeRule",
                "events:ListRuleNamesByTarget",
                "events:ListRules",
                "events:ListTargetsByRule"
            ],
            "Resource": [
                "arn:aws:events:*:*:rule/*"
            ]
        }
    ]
}
```

Replace the placeholder values with your own values.

#	Placeholder	Description	Example
1	*bucket*	The name of your S3 bucket	my-bucket
2	*region*	The AWS Region of your bucket	us-east-1
3	*accountId*	Your AWS account ID	123456789012

- A trust policy that allows S3 Files to assume the IAM role. Add the following trust policy to the IAM role to allow the S3 Files service to assume it. Replace *accountId* and *region* with your values.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowS3FilesAssumeRole",
            "Effect": "Allow",
            "Principal": {
                "Service": "elasticfilesystem.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "accountId"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3files:region:accountId:file-system/*"
                }
            }
        }
    ]
}
```

IAM role for attaching your file system to AWS compute resources

Your compute resources on which you mount an S3 file system must have an IAM role attached (for example, an EC2 instance profile) with policies that allow your compute resource to interact with your S3 file system and your source S3 bucket. You must also make sure that the bucket policies of your source bucket don't deny access from your compute resource.

Add the following two policies to the IAM role attached to your compute resource:

- **Permissions for the compute resource to connect to and interact with S3 file systems**

  The IAM role must include permissions for the mount helper to connect to and interact with S3 file systems. You can attach an AWS managed policy such as `AmazonS3FilesClientFullAccess` managed policy if you want to grant the compute resource full read and write access to your S3 file system or the `AmazonS3FilesClientReadOnlyAccess` for read-only access. You can also attach the `AmazonElasticFileSystemsUtils` managed policy if you want to enable Amazon CloudWatch monitoring. For more information and a complete list of available managed

policies for S3 Files, see AWS managed policies for Amazon S3 Files. You can also provide these permissions by adding individual IAM permissions such as `s3files:ClientMount` or `s3files:ClientWrite` (not required for read-only connections) to the IAM role of your compute resource.

- **An inline policy that grants the compute resource read access to S3 objects**

  Add the following inline policy to the IAM role. This policy lets the compute resource read objects directly from the linked S3 bucket in the same account. This improves read performance for large I/O operations. Replace *`bucket`* with your S3 bucket name, or with your bucket name and a prefix to scope access only to the objects that the compute resource needs.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ObjectReadAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::bucket/*"
        },
        {
            "Sid": "S3BucketListAccess",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::bucket"
        }
    ]
}
```

> **POSIX permissions don't apply to direct S3 access**
>
> This policy grants direct `s3:GetObject` access to the bucket. Reads made directly against Amazon S3 are governed by IAM and bucket policies, not by POSIX permissions. For more information, see the section called “POSIX permissions and direct S3 access”.

Security groups

Once your file system and mount targets are created, you must configure the right security groups to start using your file system. Security groups on both the compute resource and the mount target must allow the required traffic as shown in the table below:

Security group	Rule type	Protocol	Port	Source/destination
EC2 Instance	Outbound	TCP	2049	Mount target security group
Mount Target	Inbound	TCP	2049	EC2 instance security group

Tutorial: Getting started with S3 Files

In this tutorial, you create an S3 file system and mount it on an EC2 instance. You then test basic file operations. You can use either the S3 console or the AWS CLI to get started with S3 Files.

Getting started with S3 Files using the AWS Console

The S3 Files workflow on S3 Console consists of the following steps:

- Create your S3 file system.
- Mount the file system on your EC2 instance and run file system operations.

Prerequisites

Before getting started, make sure you have the following:

- You have completed the AWS account and compute setup.
- You are set up with Amazon EC2 and are familiar with launching EC2 instances. For more information, see Get started with Amazon EC2 in the *Amazon EC2 User Guide*. For this tutorial, use the default VPC for your EC2 instance.
- You have an IAM role for attaching your file system to AWS compute resources attached to your EC2 instance so that it can interact with your S3 file system and your S3 bucket.

Step 1: Create your S3 file system

- Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
- In the navigation bar, verify you are in the AWS Region where your EC2 instance and S3 bucket is.
- In the left navigation pane, choose **General purpose buckets**.
- Select the bucket for which you want to create a file system.
- Choose the **File systems** tab.
- Choose **Create file system**.
- Review and confirm your VPC. For this tutorial, use your default VPC.
- Choose **Create**.

When you create a file system using the AWS Management Console, S3 Files automatically creates one mount target in every Availability Zone in your default VPC and one access point for the file system. This can take a few minutes. Your file system will become available for the next step once all the resources are created.

Step 2: Mount the file system on your EC2 instance

- On the file system **Overview** page, choose **Attach** under **Attach to an EC2 instance**. This will open a new page to mount your file system on an EC2 instance.
- Select your desired EC2 instance from the dropdown **Available EC2 instances**.
- Enter a path on your EC2 instance where you want to mount the file system. For example, `/mnt/s3files/`.
- Make sure you have configured the right Security groups on your EC2 instance and the mount target to allow the required traffic to flow.
- Make sure you have the right IAM role with required permissions attached to your EC2 instance so that it can interact with your S3 file system and your S3 bucket. For more information, see IAM role for attaching your file system to AWS compute resources. For this tutorial, you can consider giving the client full access by adding the managed policy `AmazonS3FilesClientFullAccess` to EC2 instance's IAM role.
- Follow the attach instructions displayed on the page to open CloudShell, mount your file system, and run basic file system operations.

Getting started with S3 Files using the AWS CLI

The S3 Files workflow on AWS CLI consists of the following steps:

1. Create your file system.
2. Create mount targets for your file system.
3. Mount the file system on your EC2 instance using a mount target.
4. Test file operations such as listing a directory, writing text to a file, reading a file, and copying a file. Then verify that your changes reflect in your S3 bucket.

Prerequisites

Before getting started, make sure you have the following:

- You have installed and configured the AWS CLI. For more information, see Installing or updating to the latest version of the AWS CLI.
- You have completed all the prerequisites described in Prerequisites for S3 Files.
- You are set up with Amazon EC2 and are familiar with launching EC2 instances. You need an AWS account, a user with administrative access, a key pair, and a security group. For more information, see Get started with Amazon EC2 in the *Amazon EC2 User Guide*.

Step 1: Create your S3 file system

Connect to your EC2 instance. Run the `create-file-system` command to create a file system.

```
aws s3files create-file-system --region aws-region --bucket bucket-arn --role-arn iam-role
```

Replace the following with your desired values:

- *`aws-region`* : The AWS Region of your bucket. For example, `us-east-1`.
- *`bucket-arn`* : The ARN of your S3 bucket.
- *`iam-role`* : ARN of the IAM role that S3 Files assumes to read from and write to your S3 bucket. Make sure you have added the right permissions to this IAM role. For more information, see IAM role for accessing your bucket from the file system.

After successfully creating the file system, S3 Files returns the file system description as JSON. Note down the file system ID for the next step.

Step 2: Create mount targets

A mount target provides network access to your file system in your VPC within a single Availability Zone. You need a mount target to access your file system from compute resources. You can create a maximum of one mount target per Availability Zone. We recommend creating a mount target in every Availability Zone you operate in.

Run the following `create-mount-target` command to create a mount target for your file system. You must make sure the *`subnet-id`* is in the same VPC as your EC2 instance. You must create your mount target in the same Availability Zone as your EC2 instance.

```
aws s3files create-mount-target --region aws-region --file-system-id file-system-id --
subnet-id subnet-id
```

Here *`file-system-id`* is the file system ID that you received in the response of `create-file-system` command. Mount targets can take up to ~5 minutes to create.

Step 3: Mount the file system on your EC2 instance

Before mounting your file system, make sure you have configured the right Security groups on your compute resource and the mount target to allow the required traffic to flow. For more details on security groups, visit the VPC user guide.

Run the following commands to mount your file system:

- Create a directory `/mnt/s3files` that you will use as the file system mount point using the following command:

  ```
  sudo mkdir /mnt/s3files
  ```

- Mount the file system:

  ```
  sudo mount -t s3files file-system-id:/ /mnt/s3files
  ```

If you don't have your file system ID, you can find it by running the following:

```
aws s3files get-file-system --region aws-region --file-system-id file-system-id
```

Step 4: Test file operations

Test basic file operations on your mounted file system as follows:

- Change to the directory you mounted:

  ```
  cd /mnt/s3files
  ```

- You can list the contents of your directory to check that the contents of your source bucket or prefix got imported. Synchronization typically occurs within seconds, but may take longer, especially for the first file. If your bucket is empty, the command below will also return an empty result.

  ```
  ls
  ```

- You can also test other file operations:
  - Create a file:

    ```
    echo "Hello, S3 Files!" > test.txt
    ```

  - Read the file:

    ```
    cat test.txt
    ```

  - Create a directory:

    ```
    mkdir test-directory
    ```

  - Copy the file to the directory:

    ```
    cp /mnt/s3files/test.txt /mnt/s3files/test-directory/
    ```

You can then go to your S3 bucket and check that the directory `test-directory` reflects in your bucket. Note that it may take ~1 minute to synchronize changes back to your S3 bucket.

Mounting your S3 buckets on compute resources

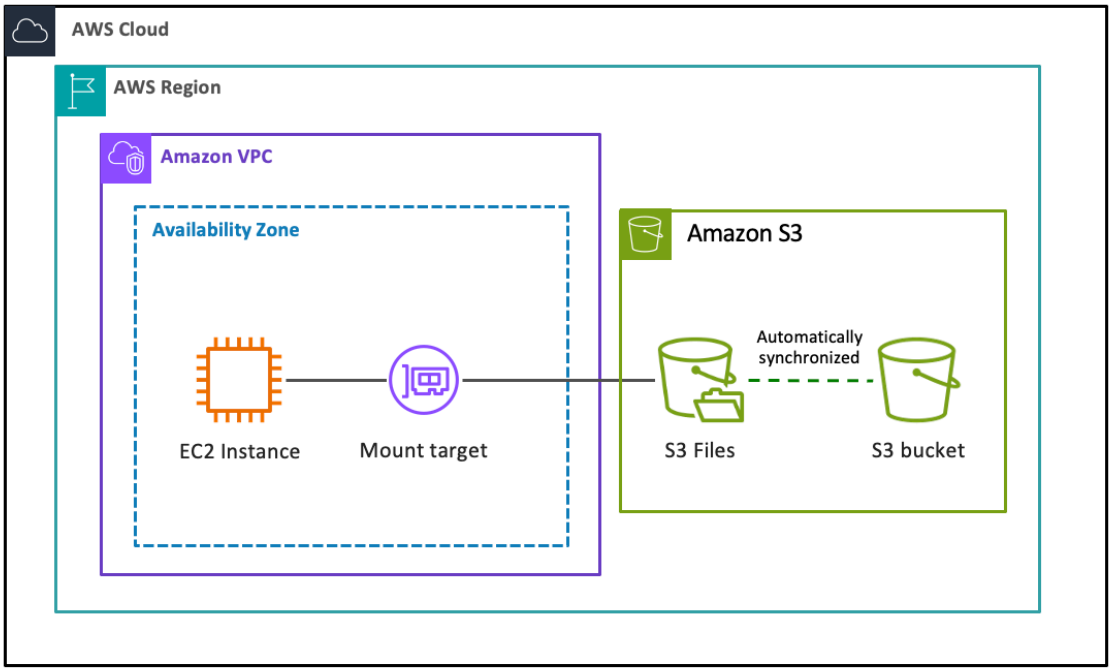
You can mount an S3 file system on compute resources to access your S3 data as files. Your compute resource must run in the same Amazon Virtual Private Cloud (Amazon VPC) as the S3 file system. All compute resources communicate with the file system through mount targets on NFS port 2049.

S3 Files supports the following compute environments:

- Amazon Elastic Compute Cloud (Amazon EC2) instances
- AWS Lambda functions
- Amazon Elastic Kubernetes Service (Amazon EKS) clusters
- Amazon Elastic Container Service (Amazon ECS) clusters

Mounting S3 file systems on Amazon EC2

To mount S3 file systems on an EC2 instance, you must use the S3 Files mount helper. The mount helper helps you mount your S3 file systems on EC2 instances running the supported distributions. When mounting a file system, the mount helper defines a new network file system type, called `s3files`, which is fully compatible with the standard `mount` command in Linux. The mount helper also supports mounting an S3 file system at instance boot time automatically by using entries in the `/etc/fstab` configuration file on EC2 Linux instances. The mount helper is part of the open-source collection of tools that is installed when you install the S3 Files client (amazon-efs-utils).



Prerequisites to mount on EC2 instances

- Your EC2 instance runs one of the following supported Linux distributions. Windows and macOS are not supported.
  - Amazon Linux 2
  - Amazon Linux 2023
  - Red Hat Enterprise Linux (RHEL) 8
  - Red Hat Enterprise Linux (RHEL) 9
  - Ubuntu 20.04
  - Ubuntu 22.04
  - Ubuntu 24.04
  - openSUSE Leap
  - SUSE Linux Enterprise Server (SLES) 15
- You have an S3 file system with at least one mount target available.
- Your EC2 instance is in the same Availability Zone as the mount target that you will use to mount your file system.
- An IAM instance profile is attached to the EC2 instance with the required permissions for S3 Files. For details, see IAM role for attaching your file system to AWS compute resources.

- You have configured the required Security groups.
- You have installed the amazon-efs-utils package on the EC2 instance. For more information, see S3 Files client.

How does the mount helper work?

When you issue a mount command, the mount helper performs the following actions:

- Retrieves IAM credentials from the EC2 instance profile.
- Initializes the efs-proxy process to establish a TLS-encrypted connection to the mount target.
- Starts the amazon-efs-mount-watchdog supervisor process, which monitors the health of TLS mounts. This process is started automatically the first time an S3 file system is mounted.
- Mounts the file system at the specified mount point.

The mount helper uses TLS version 1.2 to communicate with your file system. Using TLS requires certificates, and these certificates are signed by a trusted Amazon Certificate Authority. For more information on how encryption works, see Security for S3 Files.

The mount helper uses the following mount options that are optimized for S3 Files:

Option	Value	Description
`nfsvers`	4.2	NFS protocol version.
`rsize`	1048576	Sets the maximum number of bytes of data that the NFS client can receive for each network READ request to 1048576 (1 MB), the largest available, to avoid diminished performance.
`wsize`	1048576	Sets the maximum number of bytes of data that the NFS client can send for each network WRITE request to 1048576 (1 MB), the largest available, to avoid diminished performance.
`hard`	—	Sets the recovery behavior of the NFS client after an NFS request times out, so that NFS requests are retried indefinitely until the server replies, to ensure data integrity.

Option	Value	Description
`timeo`	600	Sets the timeout value that the NFS client uses to wait for a response before it retries an NFS request to 600 deciseconds (60 seconds) to avoid diminished performance.
`retrans`	2	Sets the number of times the NFS client retries a request before it attempts further recovery action to 2.
`noresvport`	—	Tells the NFS client to use a new non-privileged TCP source port when a network connection is reestablished. Using `noresvport` helps ensure that your file system has uninterrupted availability after a reconnection or network recovery event.

In addition, the mount helper automatically uses `tls` and `iam` mount options when mounting an S3 file system as S3 Files requires these options to establish a connection. This is because S3 Files always mounts a file system using TLS encryption and IAM authentication and these cannot be disabled.

How to mount your S3 file system on an EC2 instance?

- Connect to your EC2 instance through Secure Shell (SSH) or EC2 Instance Connect on EC2 Console. For more information, see Connect to your EC2 instance.
- Create a directory `/mnt/s3files` that you will use as the file system mount point using the following command:

```
sudo mkdir /mnt/s3files
```

- Mount your S3 file system:

```
FS="{YOUR_FILE_SYSTEM_ID}"
sudo mount -t s3files $FS:/ /mnt/s3files
```

- Confirm the file system is mounted.

```
df -h /mnt/s3files
```

You should see a response similar to the following:

```
Filesystem      Size  Used Avail Use% Mounted on
{s3files-dns}    8.0E  129M  8.0E   1% {path/to/mount}
```

You can also verify file system mount and inspect mount options by listing the contents of the local mount point. If the mount is successful, this command shows the mount details, including your mount options, for the specific directory.

```
findmnt -T /mnt/s3files
```

For detailed information on mount commands, visit the GitHub documentation.

You can now read and write S3 objects as files on your local mount path using standard file system operations. If you have objects in your S3 bucket then you can view them as files using the following commands.

```
ls /mnt/s3files
```

You can monitor your file system storage, performance, client connections, and synchronization errors using CloudWatch metrics.

How to mount your S3 file system on an EC2 instance using access points

When you mount a file system using an access point, the mount command includes the `access-point-id` mount option.

```
sudo mount -t s3files -o accesspoint=access-point-id file-system-id /mnt/s3files
```

where:

- *`access-point-id`* is the ID of your access point.
- *`file-system-id`* is the ID of your S3 file system.

Automatically mounting S3 file systems when your EC2 instance starts

You can configure your EC2 instance to automatically mount an S3 file system when the instance starts or restarts by updating the `/etc/fstab` file. The `/etc/fstab` file contains information about file systems and is used by the operating system to determine which file systems to mount at boot time.

> **Warning**
>
> Use the `_netdev` option, used to identify network file systems, when mounting your file system automatically. If `_netdev` is missing, your EC2 instance might stop responding. This result is because network file systems need to be initialized after the compute instance starts its networking. For more information, see Automatic mounting fails and the instance is unresponsive.

You can use the mount helper to configure an Amazon EC2 instance to automatically mount an S3 file system when the instance starts:

- Update the EC2 `/etc/fstab` file with an entry for the S3 file system.
- Attach an S3 file system when you create a new EC2 instance using the EC2 launch instance wizard.

Updating the /etc/fstab file

Perform the following steps to update the `/etc/fstab` on an EC2 Linux instance so that the instance uses the mount helper to automatically remount an S3 file system when the instance restarts.

- Connect to your EC2 instance.
- Open the `/etc/fstab` file in an editor and add the following line to the file:

```
file-system-id:/ mount-directory s3files _netdev 0 0
```

Where:

- *`file-system-id`* is the ID of your S3 file system (for example, `fs-0123456789abcdef0`).

- *mount-directory* is the mount point directory on your EC2 instance (for example, /mnt/s3files).
- _netdev specifies that the file system is a network file system, ensuring the instance waits for network availability before attempting the mount.

- Save the file and close the editor.
- Test the fstab entry by mounting all file systems in fstab:

```
sudo mount -a
```

- Verify the file system is mounted:

```
findmnt -T mount-directory
```

Using the nofail option

We recommend adding the nofail option to your fstab entry in production environments. This option allows the instance to boot even if the file system fails to mount:

```
file-system-id:/ mount-directory s3files _netdev,nofail 0 0
```

Automatic mounting with an access point

To automatically mount using an S3 Files access point, include the accesspoint option:

```
file-system-id:/ mount-directory s3files _netdev,accesspoint=access-point-id 0 0
```

Automatic mounting with a subdirectory

To automatically mount a specific subdirectory of your file system, specify the path:

```
file-system-id:/path/to/directory mount-directory s3files _netdev 0 0
```

Using the EC2 launch instance wizard

- Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
- Choose **Launch Instance**.

- Follow this documentation to launch an EC2 instance using the launch instance wizard in the AWS console. Before choosing **Launch Instance**, configure your network and add your S3 file system as shown in following steps.
- Make sure you select a subnet in your **Network settings**.
- Select the default security group to make sure that your EC2 instance can access your S3 file system. You can't access your EC2 instance by Secure Shell (SSH) using this security group. For access by SSH, later you can edit the default security and add a rule to allow SSH or a new security group that allows SSH. You can use the following settings:
  - Type: SSH
  - Protocol: TCP
  - Port Range: 22
  - Source: Anywhere 0.0.0.0/0
- Under **Storage** section, choose **File systems** and choose **S3 Files**.
- Under the file system drop down, you will see your file systems in the Availability Zone based on the subnet you selected previously in your Network settings. Choose the S3 file system that you want to mount. If you don't have any file systems, choose create a new file system to create a new one.
- Enter a local mount path on your EC2 instance where you want to mount the file system (for example, `/mnt/s3files`).
- A command will be generated to mount the file system and add it to fstab. You can choose to add the command to User data or run it manually on your EC2 instance after it is launched. Your EC2 instance will then be configured to mount the S3 file system at launch and whenever it's rebooted.
- Choose **Launch Instance**.

Mounting S3 file systems from another VPC

When you use a VPC peering connection or transit gateway to connect VPCs, Amazon EC2 instances that are in one VPC can access S3 file systems in another VPC.

A transit gateway is a network transit hub that you can use to interconnect your VPCs and on-premises networks. For more information about using VPC transit gateways, see Getting Started with transit gateways in the *Amazon VPC Transit Gateways Guide*. A VPC peering connection is a networking connection between two VPCs. This type of connection enables you to route traffic between them using private Internet Protocol version 4 (IPv4) or Internet Protocol version 6 (IPv6)

addresses. You can use VPC peering to connect VPCs within the same AWS Region or between AWS Regions. For more information on VPC peering, see What is VPC Peering? in the *Amazon VPC User Guide*.

When mounting a file system from a different VPC, you need to resolve the mount target manually. You should use the IP address of the mount targets in the corresponding Availability Zone as follows and replace the *mount-target-ip-address*, *file-system-id*, and *mount-directory* with your values.

```
sudo mount -t s3files -o mounttargetip=mount-target-ip-address file-system-id mount-directory
```

To ensure high availability of your file system, we recommend that you always use a mount target IP address that is in the same Availability Zone as your NFS client.

Alternatively, you can use Amazon Route 53 as your DNS service. In Route 53, you can resolve the mount target IP addresses from another VPC by creating a private hosted zone and resource record set. For more information on how to do so, see Working with private hosted zones in the *Amazon Route 53 Developer Guide*.

For more details on mounting from another VPC, visit the GitHub ReadMe.

Mounting S3 file systems from a different AWS Region

If you are mounting your S3 file system from another VPC that is in a different AWS Region than the file system, you will need to edit the `s3files-utils.conf` file. In `/etc/amazon/efs/s3files-utils.conf`, locate the following lines:

```
#region = us-east-1
```

Uncomment the line, and replace the value for the ID of the region in which the file system is located, if it is not in us-east-1.

Then, you need to specify the mount target IP in the mount command after changing the region in the config:

```
sudo mount -t s3files -o mounttargetip=mount-target-ip-address file-system-id mount-directory
```

Unmounting your S3 file system

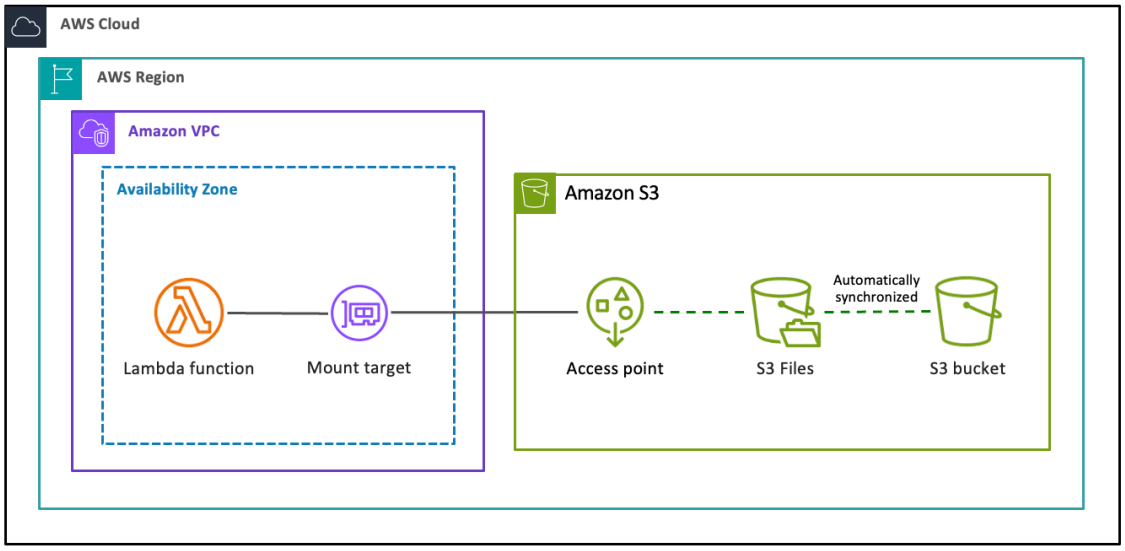
To unmount an S3 file system connected to an EC2 instance running Linux, use the `umount` command as follows:

```
umount mount-directory
```

We recommend that you do not specify any other `umount` options. Avoid setting any other `umount` options that are different from the defaults. You can verify that your S3 file system has been unmounted by running the `findmnt` command. If the unmount was successful, the `findmnt` command on your mount directory will yield no output.

Mounting S3 file systems on AWS Lambda functions

While AWS Lambda functions provide an ephemeral local storage available during execution, many serverless workloads, such as machine learning inference, data processing, and content management, require access to large reference datasets, shared files, or persistent storage. By attaching an S3 file system to your Lambda function, you can easily share data across function invocations, read large reference data files, and write function output to a persistent and shared store, all through a local mount path.



Prerequisites

Before you mount an S3 file system on a Lambda function, make sure that you have the following:

- **File system, mount targets, and access point** — The S3 file system, at least one mount target, and one access point must be available. If you create a file system using the AWS Management Console, S3 Files automatically creates one mount target in every Availability Zone in your default VPC and one access point (UID/GID 1000/1000 and `/Lambda` as the access point scope) for the file system.
- **Lambda function** — A Lambda function with an execution role that has access to mount the file system. See Execution role and user permissions in the *AWS Lambda User Guide*.
- **VPC** — The Lambda function must be in the same VPC as your mount target. The subnets you assign to your Lambda function must be in the Availability Zone that has a mount target.
- You have configured the required Security groups.

How to mount your S3 file system on a Lambda function

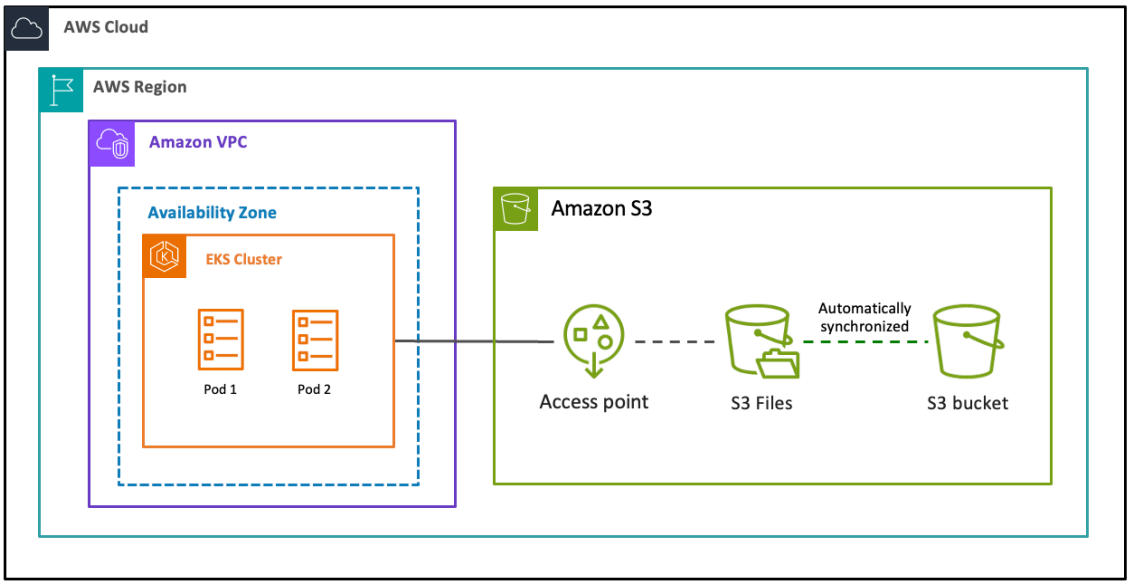
- On the S3 Console, choose **File systems** in the left navigation pane.
- Select the file system you want to mount on your Lambda function.
- In the **Overview** tab, choose **Attach** under **Attach to a Lambda function**.
- Select an available Lambda function from the drop down. The available list only shows functions within the same VPC and subnets where you have a mount target.
- Specify the local mount path.
- If you have more than one access points, select an access point.
- Choose **Attach**. Your file system will now be attached the next time you invoke your Lambda function.

For more details, see Configuring Amazon S3 Files access with AWS Lambda.

You can monitor your file system storage, performance, client connections, and synchronization errors using Amazon CloudWatch.

Mounting S3 file systems on Amazon EKS

You can attach an S3 file system to an Amazon EKS cluster by using the Amazon EFS Container Storage Interface (CSI) driver. The driver supports both dynamic and static provisioning. To set this up, install the efs-csi-driver, which is the CSI driver for both Amazon EFS and S3 Files.



Prerequisites

Before you mount an S3 file system on an EKS cluster, make sure that you have the following:

- You have an S3 file system that has at least one mount target available.
- You have configured the required Security groups.
- Your EKS cluster must be in the same VPC as your mount target.
- The Amazon EFS CSI driver needs AWS Identity and Access Management (IAM) permissions to connect to and interact with S3 file systems. For details, see IAM role for attaching your file system to AWS compute resources.
- AWS suggests using EKS Pod Identities. For more information, see Overview of setting up EKS Pod Identities.
- For information about IAM roles for service accounts and setting up an IAM OpenID Connect (OIDC) provider for your cluster, see Create an IAM OIDC provider for your cluster.
- The `kubectl` command line tool is installed on your device or AWS CloudShell. The version must be within one minor version of the Kubernetes version of your cluster. For example, if your cluster version is 1.29, you can use `kubectl` version 1.28, 1.29, or 1.30 with it. To install or upgrade `kubectl`, see Set up kubectl and eksctl.

How to mount your S3 file system on an EKS cluster

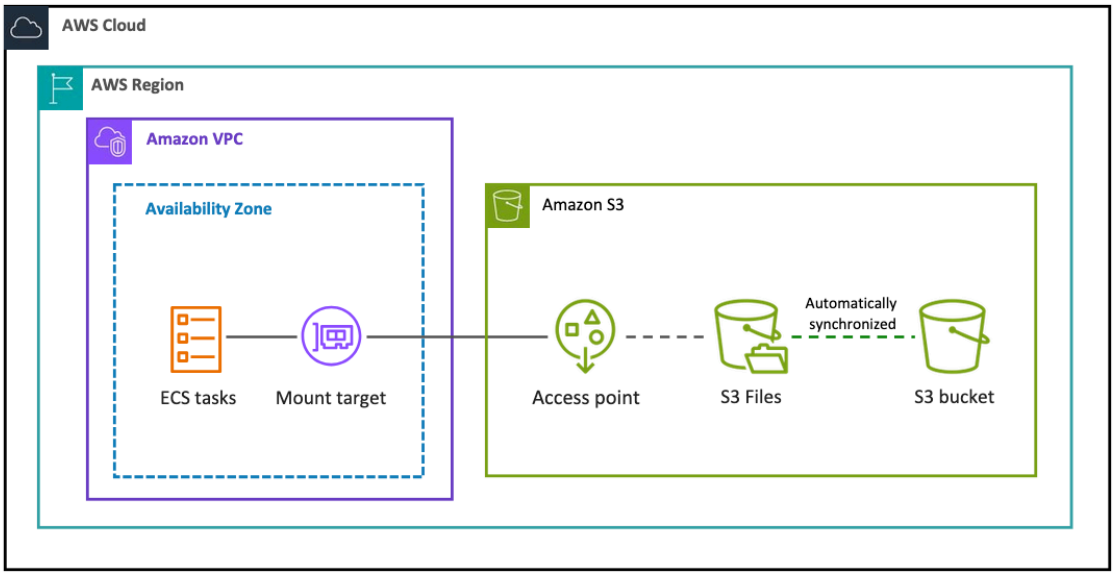
The Amazon EFS CSI driver requires IAM permissions to interact with your file system. Create an IAM role and attach the `AmazonS3FilesCSIDriverPolicy` managed policy to it. Then add the EFS CSI driver to your EKS cluster and specify the IAM role. You can use the AWS Management Console or the AWS API. For details, see Using S3 file system storage with Amazon EKS.

You can also use S3 file systems with AWS Batch on Amazon EKS. To attach an S3 file system volume to your AWS Batch job, use Amazon EKS pods with a persistent volume claim. For more details, see persistentVolumeClaim section of Register Job Definitions and EKS Persistent Volume Claim pages of the *AWS Batch API Reference Guide*.

You can monitor your file system storage, performance, client connections, and synchronization errors using Amazon CloudWatch.

Mounting S3 file systems on Amazon ECS

You can attach an S3 file system to an Amazon ECS task definition and then deploy the task to access your S3 data from your containers.



In Amazon ECS, S3 Files volume support is available for AWS Fargate and ECS Managed Instances at General Availability. S3 Files volumes are not supported on the Amazon EC2 launch type. If you configure an S3 Files volume in a task definition and attempt to run it on the EC2 launch type, the task will fail.

Prerequisites

Before you attach an S3 file system to an ECS task, make sure that you have the following:

- You have an S3 file system with at least one mount target in available state.
- The ECS task must be in the same VPC as the mount target.
- Add the permissions to your ECS task IAM role to access S3 file systems. For details, see IAM role for attaching your file system to AWS compute resources.
- You have configured the required Security groups.

How to mount your S3 file system on an ECS task

- On the S3 Console, choose **File systems** in the left navigation pane.
- Select the file system you want to mount.
- In the **Overview** tab, choose **Attach** under **Attach to an ECS task**.
- Select your desired ECS task definition from the drop down.
- Specify the local mount path.
- You can optionally specify an access point, a root directory, and a transit encryption port.
- Once the file system is attached in the task definition, you can start a task using this task definition in following ways:
  - You can deploy the task as a standalone, one-time run. For details, see Running an application as an Amazon ECS task in the *Amazon ECS Developer Guide*.
  - You can also deploy the task definition as a service. For details, see View service history using Amazon ECS service deployments in the *Amazon ECS Developer Guide*.

For details, see Using S3 file system storage with Amazon ECS.

You can monitor your file system storage, performance, client connections, and synchronization errors using Amazon CloudWatch.

Tutorial: Mount an S3 file system across accounts

You can access an S3 file system from compute resources in a different AWS account than the file system owner account. This is called cross-account access. With cross-account access, teams in different accounts can share the same file system without duplicating data.

Cross-account access requires the following:

- Network access on port 2049
- DNS that resolves the file system mount target
- S3 Files permissions that allow access from the remote account

In this tutorial, you configure cross-account access from an Amazon EC2 instance using the AWS Management Console. Account A owns the S3 file system, and Account B owns the EC2 instance that connects to it.

Prerequisites

Before you begin, make sure that you have the following:

- An S3 file system with mount targets in Account A. For more information, see Getting started with S3 Files.
- An EC2 instance running in Account B with amazon-efs-utils v3.0.0 or later.
- An IAM role for attaching your file system to AWS compute resources attached to your EC2 instance.
- A file system policy that grants Account B permissions for `s3files:ClientMount` and `s3files:ClientWrite`.
- Non-overlapping VPC CIDRs in Account A and Account B.
- Both accounts meet all S3 Files prerequisites. For more information, see S3 Files prerequisites.

Example values used in this tutorial

This tutorial uses the following example values. Replace them with your own values.

Item	Value
Account A (S3 Files account)	`111111111111`
Account B (EC2 instance account)	`222222222222`
S3 file system ID	`fs-0123456abcdef0189`
Region	`us-east-1`

Item	Value
S3 Files DNS name	*{az_id}*.fs-0123456abcdef0189.s3files.us-east-1.on.aws
EC2 instance role (Account B)	ec2-instance-role

Step 1: Record file system network details

In Account A, record the following network details from your file system.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **General purpose buckets**, then choose the bucket linked to your S3 file system.
3. Choose the **File systems** tab, choose your file system, then choose the **Mount targets** tab.
4. Verify that mount targets exist in each Availability Zone you plan to connect from.
5. Record the following values for use in later steps:
   - VPC ID of the file system
   - AZ IDs (for example, `use1-az1`, `use1-az2`)
   - IPv4 address of each mount target (mapped to each AZ ID)
   - Security group ID of the mount targets

Step 2: Create a VPC peering connection

> ⚠ **CIDR ranges must not overlap**
>
> VPC peering requires that your CIDR ranges do not overlap. Verify this before proceeding.

1. In both accounts, verify that the VPC has the following DNS settings enabled. In the VPC console, choose your VPC, then choose **Edit VPC settings**:
   - **DNS resolution (enableDnsSupport):** Enabled
   - **DNS hostnames (enableDnsHostnames):** Enabled
2. In Account B, open the VPC console, choose **Peering Connections**, then choose **Create peering connection**. Enter the following values:

- **VPC ID (Requester):** VPC of the EC2 instance (Account B)
- **Account:** Another Account, then enter `111111111111`
- **Region:** `us-east-1`
- **VPC ID (Accepter):** VPC of the S3 file system (Account A)

3. In Account A, open the VPC console, choose **Peering Connections**, find the pending request, then choose **Actions**, **Accept request**. Verify that the status changes to **Active**.

Step 3: Add routes

In Account B (EC2 instance account):

1. Open the VPC console, choose **Route Tables**, and select the route table for the EC2 instance's subnet.
2. Choose **Edit routes**, then choose **Add route**:
   - **Destination:** CIDR of the S3 Files VPC in Account A
   - **Target:** Select **Peering Connection**, then select the peering connection you created in Step 2

In Account A (S3 Files account):

1. Open the VPC console, choose **Route Tables**, and select the route table for the S3 Files VPC.
2. Choose **Edit routes**, then choose **Add route**:
   - **Destination:** CIDR of the EC2 instance VPC in Account B
   - **Target:** Select **Peering Connection**, then select the peering connection you created in Step 2

Step 4: Configure security groups

The mount target security group in Account A must allow inbound NFS traffic from the EC2 instance in Account B. The EC2 instance security group in Account B must allow outbound NFS traffic to the mount target.

1. In the VPC console, choose **Security Groups**, then select the mount target security group you noted in Step 1.
2. Choose **Edit inbound rules** and add the following rule:
   - **Type:** NFS

- **Port:** 2049
- **Source:** Choose **Custom**, then enter the EC2 security group ID from Account B

3. Verify that the EC2 instance security group in Account B allows outbound TCP on port 2049.

Security Group	Rule Type	Protocol	Port	Source/Destination
EC2 instance (Account B)	Outbound	TCP	2049	Mount target security group (Account A)
Mount target (Account A)	Inbound	TCP	2049	EC2 instance security group (Account B)

Step 5: Add a file system policy

In Account A, add a file system policy that grants the Account B role permission to mount. S3 Files evaluates this policy on every mount request. The mount fails without it, regardless of network or DNS configuration.

1. Open the Amazon S3 console, choose your bucket, then choose the **File systems** tab.
2. Select your file system, then choose **Edit file system policy**.
3. Add a policy that grants the Account B role permission to mount. The following example policy grants `s3files:ClientMount` and `s3files:ClientWrite` to the EC2 instance role in Account B:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowCrossAccountMount",
            "Effect": "Allow",
            "Principal": { "AWS": "arn:aws:iam::222222222222:role/ec2-instance-
role" },
            "Action": [
                "s3files:ClientMount",
                "s3files:ClientWrite"
            ],
```

```
            "Resource": "arn:aws:s3files:us-east-1:111111111111:file-system/
fs-0123456abcdef0189"
        }
    ]
}
```

Step 6: Create Route 53 hosted zones

In Account B, create private hosted zones so the EC2 instance can resolve the file system DNS names. S3 Files uses DNS names in the following format to route mount requests to the correct Availability Zone:

```
{az_id}.{fs_id}.s3files.{region}.on.aws
```

Create one private hosted zone for each AZ ID that has a mount target. For each AZ ID:

1. Open the Route 53 console, choose **Hosted Zones**, then choose **Create hosted zone**. Enter the following values:
   - **Domain name:** *{az_id}*`.fs-0123456abcdef0189.s3files.us-east-1.on.aws`
   - **Type:** Private hosted zone
   - **Region:** `us-east-1`
   - **VPC ID:** VPC of the EC2 instance in Account B
2. Choose **Create hosted zone**.
3. Choose **Create record** and enter the following values:
   - **Record name:** (leave blank)
   - **Record type:** A
   - **Value:** IPv4 address of the mount target for that AZ ID (from Step 1)
4. Choose **Create records**.
5. Repeat for all AZ IDs that you plan to connect from.

Example:

Hosted Zone Domain Name	A Record Value
use1-az1.fs-0123456abcdef0189.s3files.us-east-1.on.aws	Mount target IP for use1-az1
use1-az2.fs-0123456abcdef0189.s3files.us-east-1.on.aws	Mount target IP for use1-az2
use1-az4.fs-0123456abcdef0189.s3files.us-east-1.on.aws	Mount target IP for use1-az4

Step 7: Configure IAM permissions

Attach an IAM role to the EC2 instance in Account B. The role must include the following permissions to mount and access the file system:

- **The `AmazonS3FilesClientFullAccess` managed policy (or equivalent)** — This policy grants the EC2 instance permission to connect to and interact with S3 Files file systems.
- **An inline policy granting S3 object access to the cross-account bucket** — This policy grants the EC2 instance permission to read objects directly from the S3 bucket in Account A.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ObjectReadAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": "arn:aws:s3:::BUCKET_NAME_ACCOUNT_A/*"
        },
        {
            "Sid": "S3BucketListAccess",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::BUCKET_NAME_ACCOUNT_A"
        }
```

```
    ]
}
```

Replace *BUCKET_NAME_ACCOUNT_A* with the name of the S3 bucket in Account A.

> **Note**
>
> For the complete list of required base policies for your compute resource, see IAM role for attaching your file system to AWS compute resources.

Step 8: Mount the file system

Connect to your EC2 instance in Account B and run the following commands.

1. Create a mount point:

```
sudo mkdir /mnt/s3files
```

2. Mount the file system:

```
sudo mount -t s3files fs-0123456abcdef0189 /mnt/s3files
```

3. Verify the mount:

```
df -h /mnt/s3files
ls /mnt/s3files
```

Troubleshooting

Use the following information to resolve common issues when mounting an S3 file system across accounts.

Mount fails with a DNS resolution error

Verify that both VPCs have `enableDnsSupport` and `enableDnsHostnames` enabled. Verify that the Route 53 private hosted zones are associated with the correct VPC in Account B and that A records point to the correct mount target IP addresses.

Mount hangs or times out

Verify that the VPC peering connection status is **Active**. Verify that route tables in both accounts have entries for the peer VPC CIDR pointing to the peering connection. Verify that the mount target security group allows inbound TCP on port 2049 from the EC2 instance security group.

Permission denied on file operations

Verify that the EC2 instance IAM role has both the `AmazonS3FilesClientFullAccess` managed policy and the inline S3 object access policy attached. Verify that the file system policy in Account A grants `s3files:ClientMount` and `s3files:ClientWrite` to the Account B role.

Creating and managing S3 Files resources

This page describes how to create, configure, and manage S3 Files resources. To manage your resources using the AWS CLI, see S3 Files API reference.

File systems

A shared file system linked to your S3 bucket. It stores a fraction of your actively used S3 data as files and directories so that your applications and users can benefit from low-latency performance. You can access your data using standard file system operations, including reading, writing, and locking files.

- Creating file systems
- Deleting file systems

Mount targets

A mount target provides network access to your file system within a single Availability Zone in your VPC. You need at least one mount target to access your file system from compute resources, and you can create a maximum of one mount target per Availability Zone. We recommend creating one mount target in each Availability Zone you operate in so that your compute resources always have a local network path to the file system, improving both availability and latency. When you create a file system using the AWS Management Console, S3 Files automatically creates one mount target in every Availability Zone in your default VPC.

- Creating mount targets
- Managing mount targets

- Deleting mount targets

File system policies

A file system policy is an optional IAM resource policy that you can create for your S3 file system to control NFS client access to the file system.

- Creating file system policies
- Deleting file system policies

Access points

Access points are application-specific entry points to a file system that simplify managing data access at scale for shared datasets. You can use access points to enforce user identities and permissions for all file system requests that are made through the access point. Additionally, access points can restrict clients to only access data within a specified root directory and its subdirectories. When you create a file system using the AWS Management Console, S3 Files automatically creates one access point for the file system.

A file system can have a maximum of 25,000 access points. For more information, see Unsupported features, limits, and quotas.

- Creating access points for an S3 file system
- Deleting access points for an S3 file system

Tags

Tags are key-value pairs that you define and associate with your S3 Files resources to help organize, identify, and manage them.

- Tagging S3 Files resources

CloudFormation template

You can also use CloudFormation templates to create and manage S3 Files resources. See Amazon Simple Storage Service Files (Amazon S3 Files) in the *AWS CloudFormation User Guide* for all available S3 Files resource types.

Topics

- Creating file systems
- Deleting file systems
- Creating mount targets

- Managing mount targets
- Deleting mount targets
- Creating file system policies
- Deleting file system policies
- Creating access points for an S3 file system
- Deleting access points for an S3 file system
- Tagging S3 Files resources

Creating file systems

You can create file systems by using the AWS Console, the AWS Command Line Interface (AWS CLI), or the Amazon S3 API for any existing or new S3 general purpose bucket. For information on creating a new bucket, see Creating a general purpose bucket.

Required IAM permissions for creating file systems

When you create an S3 file system, you must specify an IAM role that S3 Files assumes to read from and write to your S3 bucket. This role allows S3 Files to synchronize changes between your file system and your S3 bucket. When you create a file system using the AWS Console, S3 Files automatically creates this IAM role with the required permissions. If you are using the AWS CLI or S3 API, see IAM role for accessing your bucket from the file system.

For more information about managing permissions for API operations, see How S3 Files works with IAM.

Status of a file system

A file system can have one of the status values described in the following table that you can get using the `get-file-system` command.

File system state	Description
AVAILABLE	The file system is in a healthy state, and is reachable and available for use.
CREATING	S3 Files is in the process of creating the new file system.

File system state	Description
DELETING	S3 Files is deleting the file system in response to a user-initiated delete request.
DELETED	S3 Files has deleted the file system in response to a user-initiated delete request.
ERROR	The file system is in a failed state and is unrecoverable. To access the file system data, restore a backup of this file system to a new file system. Check the StatusMessage field for information about the error.
MISCONFIGURED	A permissions issue prevents S3 Files from synchronizing changes between the file system and its S3 bucket. For more information, see File system status is MISCONFIGURED.

Note

S3 Files returns an error when you attempt to create a file system scoped to a prefix with a large number of objects. This error alerts you that large recursive rename or move operations may impact file system performance and increase S3 request costs, as every file requires separate copy and delete requests to your S3 bucket. If you still want to create a file system scoped to that prefix, you can add the `--AcceptBucketWarning` parameter.

Using the S3 console

This section explains how to use the Amazon S3 console to create a file system for S3 Files.

- Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
- In the navigation bar, verify you are in your desired AWS Region.
- In the left navigation pane, choose **File systems**.
- Select **Create file system**.
- On the create page, choose the S3 bucket or prefix to create your file system from. You can enter the S3 URI directly (for example, `s3://bucket-name/prefix`) or choose **Browse S3** to navigate to and select your bucket or prefix.

- Select a VPC for your file system. S3 Files selects your default VPC automatically. This is the VPC where your compute resources connect to your file system. To use a different VPC, choose one from the dropdown.
- Select **Create** and wait for the status of your file system to become `Available`.

Default settings on AWS Management Console

S3 Files will create your file system with the following configuration:

- **Encryption** — S3 Files sets the encryption configuration from the source S3 bucket and applies it to data at rest in your file system.
- **IAM role** — S3 Files creates a new IAM role that it assumes to manage the data synchronization between your file system and bucket.
- **Mount targets** — S3 Files automatically creates one mount target in every Availability Zone in the VPC you choose.
- **Access point** — S3 Files creates one access point for the file system.

Using the AWS CLI

When you're using the AWS CLI, you create these resources in order. First, you create a file system. Then, you can create mount targets and any additional optional tags for the file system by using corresponding AWS CLI commands.

The following `create-file-system` example command shows how you can use the AWS CLI to create a file system for S3 Files.

```
aws s3files create-file-system --region aws-region --bucket s3-bucket-arn --client-
token idempotency-token --role-arn iam-role
```

Replace the following with your desired values:

- *`aws-region`* : The AWS Region of your bucket. For example, `us-east-1`.
- *`bucket-arn`* : The ARN of your S3 bucket.
- *`idempotency-token`* : An idempotency token. This is optional.
- *`iam-role`* : ARN of the IAM role that S3 Files assumes to read from and write to your S3 bucket. Make sure you have added the right permissions to this IAM role. For more information, see IAM role for accessing your bucket from the file system.

After successfully creating the file system, S3 Files returns the file system description as JSON.

Deleting file systems

When you delete a file system, the file system, its data, and its configuration are permanently removed. Make sure no applications are actively using the file system before deletion to avoid service disruption. Before deletion, you must delete all mount targets and access points associated with the file system first. For more information, see Deleting mount targets and Deleting access points for an S3 file system.

When you delete a file system, S3 Files checks whether all changes in your file system have been synchronized with your linked S3 bucket. If there are changes that have not yet been synchronized, S3 Files returns an error and the deletion does not proceed. This ensures that all your data is safely stored in your S3 bucket before the file system is deleted. If you want to proceed with deletion and accept that any unsynchronized changes will be lost, you can retry the delete request with the 'force delete' option. In the AWS CLI, add the `--ForceDelete` flag to your delete API call. On the AWS Console, choose **Force** button in the error message that appears when you delete a file system while unsynced changes are present.

Using the S3 console

This section explains how to use the Amazon S3 console to delete a file system for S3 Files.

- Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
- In the navigation bar, verify you are in the AWS Region of the file system that you want to delete.
- In the left navigation pane, choose **General purpose buckets**.
- Choose a general purpose bucket your file system is attached to.
- Select the **File systems** tab and select the file system you wish to delete.
- Choose **Delete**.
- In the confirmation window, type `confirm`.

Using the AWS CLI

The following `delete-file-system` example command shows how you can use the AWS CLI to delete a file system for S3 Files.

```
aws s3files delete-file-system --file-system-id file-system-id
```

Creating mount targets

You need a mount target to access your file system from compute resources and you can create a maximum of one mount target per Availability Zone. We recommend creating one mount target per Availability Zone you operate in. When you create a file system using the S3 console, S3 Files automatically creates one mount target in every Availability Zone in your default VPC.

You can create mount targets for the file system in one VPC at a time. If you want to modify the VPC for your mount targets, you need to first delete all the existing mount targets for the file system and then create a mount target in a new VPC. If the VPC has multiple subnets in an Availability Zone, you can create a mount target in only one of those subnets. All EC2 instances in the Availability Zone can share the single mount target.

Using the S3 console

This section explains how to use the Amazon S3 console to create a mount target for S3 Files.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, verify you are in the AWS Region of the file system for which you want to create a mount target.
3. In the left navigation pane, choose **General purpose buckets**.
4. Choose a general purpose bucket your file system is attached to.
5. Select the **File systems** tab and select your desired file system.
6. Select the **Mount targets** tab and select **Create mount targets**.
7. On the Create mount target page, your default VPC will automatically be selected. Choose the Availability Zone and Subnet ID. The VPC, Availability Zone, and Subnet ID cannot be edited after mount target creation.

> **Note**
>
> The IP address type must match the IP type of the subnet. Additionally, the IP address type overrides the IP addressing attribute of your subnet. For example, if the IP address type is IPv4-only and the IPv6 addressing attribute is enabled for your subnet, network interfaces created in the subnet receive an IPv4 address from the range of the subnet. For more information, see Modify the IP addressing attributes of your subnet.

8. If you know the IP address where you want to place the mount target, then enter it in the IP address box that matches the IP address type. If you don't specify a value, S3 Files selects an unused IP address from the specified subnet.
9. Choose your security groups to associate with the mount target. See Security groups in the prerequisites to understand the security group configurations required to start using your file system.
10. Choose **Create mount target**.

Using the AWS CLI

The following `create-mount-target` example command shows how you can use the AWS CLI to create a mount target for S3 Files.

```
aws s3files create-mount-target --region aws-region --file-system-id file-system-id --
subnet-id subnet-id
```

Mount targets can take up to ~5 minutes to create.

Managing mount targets

You can add or remove security groups associated with a mount target. Security groups define inbound and outbound access. When you change security groups associated with a mount target, make sure that you authorize necessary inbound and outbound access. Doing so enables your compute resource to communicate with the file system. See Security groups in the prerequisites to understand the security group configurations required to start using your file system.

Using the S3 console

This section explains how to use the Amazon S3 console to add or remove security groups for a mount target in S3 Files.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, verify you are in the desired AWS Region where your mount target exists.
3. In the left navigation pane, choose **General purpose buckets**.
4. Choose a general purpose bucket your file system is attached to.
5. Select the **File systems** tab and select your desired file system.

6. Select the **Mount targets** tab and select the mount target that you want to edit.
7. Choose **Edit**. You will see details of your mount target.
8. Add or remove security groups from the security group drop down.
9. Choose **Save**.

Using the AWS CLI

The following `update-mount-target` example command shows how you can use the AWS CLI to add or remove security groups for a mount target in S3 Files.

```
aws s3files update-mount-target --region aws-region --mount-target-id mount-target-id
 --security-groups security-group-ids-separated-by-space
```

Deleting mount targets

When you delete a mount target, it breaks any mounts of the file system. This might disrupt compute resources and applications using those mounts. To avoid disruption, stop applications and unmount the file system before you delete the mount target.

You can delete mount targets for a file system by using the AWS Management Console, AWS CLI, or programmatically by using the AWS SDKs.

Using the S3 console

Use the Amazon S3 console to delete a mount target for S3 Files.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, verify you are in the AWS Region of the mount target that you want to delete.
3. In the left navigation pane, choose **General purpose buckets**.
4. Choose a general purpose bucket your file system is attached to.
5. Select the **File systems** tab and select your desired file system.
6. Select the **Mount targets** tab and select the mount target you wish to delete.
7. Choose **Delete**.
8. In the confirmation window, type **`confirm`** and choose **Delete**.

Using the AWS CLI

The following `delete-mount-target` command deletes a mount target for S3 Files using the AWS CLI.

```
aws s3files delete-mount-target --region aws-region --mount-target-id mount-target-id
```

Creating file system policies

You can use file system policies to grant or deny permissions for NFS clients to perform operations such as mounting, writing, and root access on your file systems. A file system either has an empty (default) file system policy or exactly one explicit policy. You can update your file system policy at any time after file system creation using the AWS Management Console, AWS CLI, or AWS SDK.

You can update a file system policy by using the Amazon S3 console, the AWS CLI, programmatically with AWS SDKs, or the S3 Files API directly. These policy changes can take several minutes to take effect. S3 file system policies have a 20,000 character limit. For more information about using an S3 file system policy, supported actions, and supported condition keys, see How S3 Files works with IAM.

Using the S3 console

This section explains how to use the Amazon S3 console to create a file system policy for S3 Files.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, verify you are in the AWS Region where your file system exists.
3. In the left navigation pane, choose **File systems**.
4. Choose your desired file system.
5. Select the **Permissions** tab and select **Edit**.
6. You can use the Policy editor to add your own file system policy.
7. After you complete editing the policy, choose **Save**.

Deleting file system policies

You can delete a file system policy using the Amazon S3 console and the AWS CLI.

Using the S3 console

This section explains how to use the Amazon S3 console to delete a file system policy for S3 Files.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, verify you are in the AWS Region where your file system exists.
3. In the left navigation pane, choose **File systems**.
4. Choose your file system.
5. Select the **Permissions** tab and select **Delete**.
6. In the confirmation window, type **`confirm`** and choose **Delete**.

Using the AWS CLI

The following `delete-file-system-policy` example command shows how you can use the AWS CLI to delete a file system policy for S3 Files.

```
aws s3files delete-file-system-policy --file-system-id file-system-id
```

Creating access points for an S3 file system

Access points are application-specific entry points to a file system that simplify managing data access at scale for shared datasets. You can use access points to enforce user identities and permissions for all file system requests that are made through the access point. Additionally, access points can restrict clients to only access data within a specified root directory and its subdirectories. When you create a file system using the AWS Management Console, S3 Files automatically creates one access point for the file system.

A file system can have a maximum of 25,000 access points. For more information, see Unsupported features, limits, and quotas. You can create access points using the Amazon S3 console, AWS CLI, or AWS SDK.

Access points for an S3 file system cannot be edited after creation. If you want to make updates, you have to delete the existing access point and create a new one.

Using the S3 console

This section explains how to use the Amazon S3 console to create an access point for an S3 file system.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, verify you are in the AWS Region of the file system for which you want to create an access point.
3. In the left navigation pane, choose **File systems**.
4. Choose your desired file system.
5. Select the **Access points** tab and select **Create access point**.
6. On the Create page, enter a **Name** for the access point.
7. (Optional) Specify a root directory path for the access point. Clients using this access point will be limited to this directory and its subdirectories. By default, S3 Files assumes the root directory for the access point to be the root directory of the file system.
8. (Optional) In the **POSIX user** panel, you can specify the full POSIX identity to use to enforce user and group information for all file operations by clients that are using the access point.
   - **User ID** – Enter a numeric POSIX user ID for the user.
   - **Group ID** – Enter a numeric POSIX group ID for the user.
   - **Secondary group IDs** – Enter an optional comma-separated list of secondary group IDs.
9. (Optional) For **Root directory creation permissions**, you can specify the permissions to use when S3 Files creates the root directory path, if specified and the root directory doesn't already exist. If the root directory already exists, including a directory imported from the linked S3 bucket, S3 Files doesn't apply these permissions. To change the directory's owner or permissions, see Setting POSIX metadata on a directory.

   > **Note**
   >
   > If you don't specify any root directory ownership and permissions, and the root directory does not already exist, S3 Files will not create the root directory. Any attempts to mount the file system by using the access point will fail.

   - **Owner user ID** – Enter the numeric POSIX user ID to use as the root directory owner.
   - **Owner group ID** – Enter the numeric POSIX group ID to use as the root directory owner group.
   - **Permissions** – Enter the Unix mode of the directory. A common configuration is 755. Ensure that the execute bit is set for the access point user so that they are able to mount.
10. (Optional) Under **Tags**, you can choose to add tags to your access point.

11. Choose **Create access point**.

Using the AWS CLI

The following `create-access-point` example command shows how you can use the AWS CLI to create an access point for an S3 file system.

```
aws s3files create-access-point --file-system-id file-system-id --root-directory root-directory --posix-user posix-user
```

For example:

```
aws s3files create-access-point --file-system-id fs-abcdef0123456789a --client-token
 010102020-3 \
  --root-directory "Path=/s3files/mobileapp/
east,CreationInfo={OwnerUid=0,OwnerGid=11,Permissions=775}" \
  --posix-user "Uid=22,Gid=4" \
  --tags Key=Name,Value=east-users
```

> **Note**
>
> If multiple requests to create access points on the same file system are sent in quick succession, and the file system is nearing the access points limit, you may experience a throttling response for these requests. This is to ensure that the file system does not exceed the access point quota.

Deleting access points for an S3 file system

When deleting an access point, make sure no applications are actively using the access point before deletion to avoid service disruption. Once deleted, the access point and its configuration are permanently removed.

Using the S3 console

This section explains how to use the Amazon S3 console to delete an access point for S3 Files.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the navigation bar, verify you are in the AWS Region of the file system which has the access point that you want to delete.
3. In the left navigation pane, choose **General purpose buckets**.
4. Choose a general purpose bucket your file system is attached to.
5. Select the **File systems** tab and select the file system you wish to use.
6. Select the **Access points** tab and select the access point you wish to delete.
7. Choose **Delete**.
8. In the confirmation window, type **`confirm`** and choose **Delete**.

Using the AWS CLI

The following `delete-access-point` example command shows how you can use the AWS CLI to delete an access point for S3 Files.

```
aws s3files delete-access-point --access-point-id access-point-id
```

Tagging S3 Files resources

To help you manage your S3 Files resources, you can assign your own metadata to each resource in the form of tags. With tags, you can categorize your AWS resources in different ways, for example, by purpose, owner, or environment. This categorization is useful when you have many resources of the same type as you can quickly identify a specific resource based on the tags that you've assigned to it. You can tag S3 file system and access point resources that already exist in your account. This topic describes tags and shows you how to create them.

Tag restrictions

The following basic restrictions apply to tags:

- Maximum number of tags per resource – 50
- For each resource, each tag key must be unique, and each tag key can have only one value.
- Maximum key length – 128 Unicode characters in UTF-8
- Maximum value length – 256 Unicode characters in UTF-8
- The allowed characters are: letters, numbers, and spaces representable in UTF-8, and the following characters: `+ - = . _ : / @`.

- Tag keys and values are case-sensitive.
- The `aws:` prefix is reserved for AWS use. If a tag has a tag key with this prefix, then you can't edit or delete the tag's key or value. Tags with the `aws:` prefix do not count against your tags per resource limit.

You can't update or delete a resource based solely on its tags; you must specify the resource identifier. For example, to delete file systems that you tagged with a tag key called `DeleteMe`, you must use the `DeleteFileSystem` action with the resource identifiers of the file system, such as the file system ID.

When you tag public or shared resources, the tags that you assign are available only to your AWS account. No other AWS account will have access to those tags. For tag-based access control to shared resources, each AWS account must assign its own set of tags to control access to the resource.

Using the S3 console

You can use the S3 Files console to manage tags on your resources.

- Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
- In the navigation bar, verify you are in your desired AWS Region.
- In the left navigation pane, choose **File systems**.
- You can specify tags for a resource when you create the resource, such as an S3 file system or an access point. Or, you can add, modify, or delete tags after creation by going to the properties of the resource.

Using the AWS CLI

If you're using the S3 Files API, the AWS CLI, or an AWS SDK, you can use the `TagResource` S3 Files API action to apply tags to existing resources. Additionally, some resource-creating actions enable you to specify tags for a resource when the resource is created, such as when you create a file system.

The AWS CLI commands for managing tags, and the equivalent S3 Files API actions, are listed in the following table.

CLI command	Description	Equivalent API operation
`tag-resource`	Add new tags or update existing tags	`TagResource`
`list-tags-for-resource`	Retrieve existing tags	`ListTagsForResource`
`untag-resource`	Delete existing tags	`UntagResource`

Understanding how synchronization works

S3 Files keeps your file system and the linked S3 bucket synchronized automatically. The data you actively use is copied to the file system, so you can read and write files using standard Linux file operations at low latency. S3 Files requires S3 Versioning to be enabled on the linked S3 bucket. When you edit files on the file system, S3 Files copies your changes back to the S3 bucket as new versions of the corresponding objects, making sure the old versions are preserved. When other applications add, modify, or delete objects in your S3 bucket, S3 Files automatically reflects those changes in your file system. When a conflict occurs due to concurrent changes to the same data in both the file system and the S3 bucket, S3 Files treats the S3 bucket as the source of truth in case of conflicts.

To optimize storage costs, S3 Files removes data you have not used recently from the file system. Your data remains durably stored in the linked S3 bucket and is fetched back onto the file system the next time you access it.

S3 bucket is accessible through the file system

After you create an S3 file system, you can mount your S3 buckets on compute resources and start accessing your S3 bucket data right away. By default, when you first access a directory by listing its contents or opening a file within it, S3 Files imports the metadata for all files in that directory, along with the data for files smaller than the import size threshold (default 128 KiB) from the S3 bucket. The first access to a directory might have higher latency, but subsequent reads and writes are significantly faster. By importing metadata upfront, S3 Files enables you to browse directory contents, view file sizes, and check permissions at low latency.

For example, suppose your S3 bucket contains a prefix `data/images/` with 1,000 objects. The first time you run `ls /mnt/s3files/data/images/`, S3 Files imports metadata for all 1,000 files

and asynchronously copies data for files below the import size threshold onto the file system. This initial listing may take several seconds, but subsequent commands such as `ls -la`, `stat`, or `cat` on individual files in that directory return at low latency.

For files larger than the import size threshold, S3 Files imports only metadata, while data is not copied to the file system and is instead read directly from the S3 bucket when you access it. You can adjust this threshold to better match your workload. For example, you can increase it to import more data up front for workloads that repeatedly access the same files and benefit from low-latency reads. For workloads that stream data sequentially, a lower threshold can be more cost effective, as the latency benefit of importing data up front is less meaningful when data is read sequentially in large chunks rather than in small, random reads. For more information, see Customizing synchronization for S3 Files.

Changes in your file system automatically reflect in your S3 bucket

When you create, modify, or delete files in the file system, S3 Files automatically copies those changes to your S3 bucket. New files become new S3 objects, changes to existing files become new object versions, and deleted files become S3 delete markers.

POSIX permissions that you set on files and directories through the file system, such as owner (UID), group (GID), and permission bits, are stored as user-defined S3 object metadata on the corresponding S3 objects. When you change permissions using `chmod`, `chown`, or `chgrp`, S3 Files exports those changes to your S3 bucket along with any data changes. When S3 Files imports objects from your S3 bucket, it reads this metadata and applies the corresponding POSIX permissions on the file system. Objects that do not have POSIX permission metadata are assigned default permissions.

When a file is modified on the file system, S3 Files waits for a period of write inactivity (60 seconds) before exporting those changes back to your S3 bucket. Rapid successive writes to the same file are captured in a single S3 PUT request rather than generating multiple object versions for each individual change. This reduces both your S3 request costs and storage costs. You can use this 60-second inactivity window as an export trigger. For example, if your application appends data to a file every 30 seconds for a total of 5 minutes, S3 Files starts the export process at minute 6. If you make additional changes, the process repeats after each 60-second period of no write activity.

Changes in your S3 bucket automatically appear in your file system

S3 Files monitors changes in your S3 bucket using S3 Event Notifications. When another application working with the S3 API adds, modifies, or deletes objects in your S3 bucket, S3 Files

automatically reflects those changes in the file system for files whose data is currently stored in the file system's high performance storage. Files whose data has been expired from the file system are not updated until the next time you access them, at which point S3 Files retrieves the latest version from the S3 bucket.

Understanding the impact of rename and move operations

Amazon S3 uses a flat storage structure where objects are identified by their key names. While S3 Files lets you organize your data in directories, S3 has no native concept of directories. What appears as a directory in your file system is a common prefix shared by the keys of the objects within the S3 bucket. Additionally, S3 objects are immutable and do not support atomic renames. As a result, when you rename or move a file, S3 Files must write the data to a new object with the updated key and delete the original. When you rename or move a directory, S3 Files must repeat this process for every object that shares that prefix. Therefore, when you rename or move a directory containing tens of millions of files, your S3 request costs and the synchronization time increase significantly.

S3 Files returns an error when you attempt to create a file system scoped to a prefix with a large number of objects such that a rename can take up to 4 hours (approximately up to 12 million objects). This error alerts you that large recursive rename or move operations may impact file system performance, as every file requires separate write and delete requests to your S3 bucket. If you still want to create a file system scoped to that prefix, you can add the `--AcceptBucketWarning` parameter.

Since S3 Files renames objects individually on the S3 bucket, both directories will be visible on the S3 bucket until the rename is fully completed. Objects written after the directory was renamed but before that rename is fully synchronized will not be moved. To simplify data reorganization work, we recommend you do not create new objects through the S3 bucket while renaming a matching directory.

For example, if you run `mv /mnt/s3files/projects/alpha /mnt/s3files/projects/beta`, the rename completes instantly on the file system. On the S3 bucket, S3 Files begins copying and deleting each object to its new key within the S3 bucket (replacing the `projects/alpha/` prefix with `projects/beta/`) and deleting the original. During this process, the S3 bucket temporarily contains objects under both `projects/alpha/` and `projects/beta/`. Once all objects have been moved, only `projects/beta/` remains.

Unused data is expired from the file system to optimize storage

S3 Files optimizes storage costs by automatically removing file data that has not been read recently from the file system. Your data remains safely stored in your S3 bucket. S3 Files only removes the copy from the file system. File metadata, such as names, sizes, and permissions, is never removed from the file system so you can continue browsing your file system at low latency.

If a file in your file system has not been read for 30 days (configurable) and its changes have already been synchronized to the S3 bucket, S3 Files removes the file data from the file system. The next time you read that file, S3 Files retrieves the latest version of the corresponding object from the S3 bucket and copies it back onto the file system.

For example, suppose you process a dataset in `/mnt/s3files/data/batch-jan.parquet` in January and do not access it again. After 30 days, S3 Files removes the file data from the file system. The file still appears in directory listings with its correct size and permissions, but the data is no longer on the file system. When you read the file again in April, S3 Files retrieves it from the S3 bucket and copies it back onto the file system. The first read may have higher latency, but subsequent reads are fast.

S3 bucket is the source of truth in case of conflicts

A conflict occurs when the same file has been modified through the file system and the corresponding S3 object has also changed before S3 Files has synchronized the file system changes back to the S3 bucket. For example, you might edit a file through your mounted file system while another application uploads a new version of the corresponding object, or deletes it, directly in the linked S3 bucket.

S3 Files detects conflicts when it attempts to synchronize your file system changes back to the S3 bucket, or when it receives an S3 event notification indicating that the object has changed. Your S3 bucket serves as the long-term store for your data, so S3 Files considers the S3 bucket as the source of truth when a conflict occurs. This provides predictable consistency, ensuring that the version in your S3 bucket always takes precedence. In case of a conflict, S3 Files moves the conflicting file from its current location in your file system to a lost and found directory and imports the latest version from the linked S3 bucket into the file system.

For example, suppose you edit `/mnt/s3files/report.csv` through the file system. Before S3 Files synchronizes your changes back to the S3 bucket, another application uploads a new version of `report.csv` directly to the S3 bucket. When S3 Files detects the conflict, it moves your

version of `report.csv` to the lost and found directory and replaces it with the version from the S3 bucket.

The lost and found directory is located in your file system's root directory under the name `.s3files-lost+found-`*`file-system-id`*. If you mount your file system through an access point that specifies a root directory, the lost and found directory is not visible from that mount because it is located above the access point's root directory. To access the lost and found directory, mount the file system without an access point or use an access point without a root directory restriction so that you can access the full file system's scope via the access point.

When S3 Files moves a file to the lost and found directory, it prepends the file name with an identifier to distinguish multiple versions of the same file that may be moved over time. Files in the lost and found directory are not copied to your S3 bucket. You can delete files and copy files from this directory, but you cannot move or rename files within it or delete the directory itself. If you want to keep your file system changes instead of the latest version in the S3 bucket, copy the file from the lost and found directory back to its original path. You can retrieve the file's original path from the extended attributes of the file in the lost and found directory. S3 Files will then copy it to your S3 bucket as a new version of the object. For more information, see Troubleshooting S3 Files.

> **Note**
>
> Conflicting files that S3 Files moves to the lost and found directory remain there indefinitely and count toward your file system storage costs. You should delete files from the lost and found directory to free up storage when they are no longer needed.

The default synchronization settings will work for most workloads for low-latency, file-based access to S3 data. For more details about how to configure these parameters, see Customizing synchronization for S3 Files.

Topics

- Customizing synchronization for S3 Files

Customizing synchronization for S3 Files

S3 Files lets you control how data flows between your file system and linked S3 bucket through a synchronization configuration. The default settings balance latency and cost for most workloads,

but you can tune them to match your access patterns. Importing more data up front reduces read latency at the cost of higher storage and write charges. Importing less data keeps storage costs low but means more reads are served from S3 with higher latency. Each configuration has two components: import data rules, which control what data is copied onto the file system and when, and expiration data rules, which control how long unused data stays on the file system. You can update these rules using the AWS Management Console or the PutSynchronizationConfiguration API.

Import data rules

Import data rules control how data is copied from your bucket to the file system. You can have a maximum of 10 import data rules per file system. Each import data rule has the following parameters:

prefix – The S3 prefix that the rule applies to. Specify an empty string ("") for the entire bucket (file system scope) or a specific prefix (for example, "data/ml/") within the file system. The prefix must end with a forward slash (/), unless specifying the entire bucket with "". You must include exactly one import rule for the root directory. Default: "" (entire bucket or file system scope).

trigger – When to import data: ON_DIRECTORY_FIRST_ACCESS or ON_FILE_ACCESS. Default: ON_DIRECTORY_FIRST_ACCESS.

- **ON_DIRECTORY_FIRST_ACCESS** – File data is imported when you first access a directory. For example, when you first access a directory by listing its contents or opening a file within it, data is imported for all immediate children files in that directory smaller than the sizeLessThan threshold. This option is useful for workloads that require low latency when first accessing files.
- **ON_FILE_ACCESS** – File data is imported only when a file is read for the first time. This option minimizes the data imported at the cost of higher latency on first read.

sizeLessThan – Maximum file size (in bytes) to automatically import. While S3 Files imports metadata for all files, it only imports data for files smaller than this threshold. Minimum: 0 bytes (no data imported, metadata will still be imported). Maximum: 52,673,613,135,872 bytes (48 TiB). Default: 131,072 bytes (128 KiB).

> **Note**
>
> Each imported file is metered as a write. A higher sizeLessThan makes more and larger files eligible for import, which increases your write charges. This effect is largest with

ON_DIRECTORY_FIRST_ACCESS, where a single directory traversal imports every eligible file. For more information about metering and pricing, see How S3 Files is metered.

Prefix matching behavior

When multiple import data rules match a file, S3 Files applies the rule with the most specific prefix. For example, assume you have three rules:

- Rule 1: prefix = "" (entire bucket), sizeLessThan = 64 KiB, trigger = ON_FILE_ACCESS
- Rule 2: prefix = "hot/", sizeLessThan = 1 MiB, trigger = ON_DIRECTORY_FIRST_ACCESS
- Rule 3: prefix = "hot/largeData/", sizeLessThan = 256 KiB, trigger = ON_DIRECTORY_FIRST_ACCESS

For a file at hot/largeData/data.txt, S3 Files applies Rule 3. For a file at hot/data.txt, S3 Files applies Rule 2. For a file at cold/data.txt, S3 Files applies Rule 1 because there is no specific rule for the cold/ prefix.

Expiration data rules

Expiration data rules control when unused data is removed from the file system to optimize storage costs. S3 Files removes data after it has not been read for a specified duration and its changes have already been synchronized to the S3 bucket. Whenever a file is read, its expiration timer resets, extending the time that data remains in the file system. You can specify the following parameter in expiration data rules:

daysAfterLastAccess – Number of days after last read when data is removed from the file system. Minimum: 1 day. Maximum: 365 days. Default: 30 days.

If you have long-running workloads that frequently access the same data, consider longer expiration periods (30–90 days). For temporary data, consider shorter periods (1–7 days).

Example configurations

General purpose file share (default configuration) – A team of developers and data scientists mounts an S3 file system to share code, configuration files, and small datasets. Most files are under 128 KiB and are read repeatedly throughout the day. The default configuration works well for this workload: ON_DIRECTORY_FIRST_ACCESS imports metadata and small file data when any

file in a directory is first accessed, which works well when files in the same directory are likely to be accessed together, such as source files in a project or configuration files in a deployment. Subsequent access by any user is fast. When a user opens a large file such as a log archive, S3 Files automatically streams it directly from S3 for high throughput. The 30-day expiration window keeps actively used files on the file system without manual cleanup.

ML training with repeated reads – A training job reads thousands of small files (<10 MiB) repeatedly across multiple epochs. To minimize latency, set a high sizeLessThan threshold (for example, 10 MiB) with ON_DIRECTORY_FIRST_ACCESS so that file data is preloaded when the training script first lists each directory. Set a short expiration (for example, 3 days) so that data is removed from the file system promptly after the training job completes.

Agentic workloads with broad file discovery – An AI agent explores a large repository of documents, code, or knowledge base files to answer queries, reading many small files once as it searches for relevant context. Set sizeLessThan to 0 so that no data is imported onto the file system. The agent can browse the full directory tree at low latency to discover files, while each file read is served directly from S3. This keeps costs low for workloads that touch many files unpredictably but rarely revisit the same file, and scales naturally as you add more agents reading in parallel.

Hot and cold prefixes – A file system contains both frequently accessed configuration files under `config/` and infrequently accessed archive data under `archive/`. Create two import rules: one for `config/` with a high sizeLessThan and ON_DIRECTORY_FIRST_ACCESS, and one for `archive/` with sizeLessThan set to 0 and ON_FILE_ACCESS. This keeps configuration files on the file system for fast access while avoiding storage costs for archive data that is rarely read.

Monitoring and auditing S3 Files

S3 Files integrates with the following AWS services to help you monitor your file systems:

Amazon CloudWatch

By default, S3 Files metric data is automatically sent to CloudWatch at 1-minute periods, unless noted differently for some individual metrics. You can also watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Amazon EC2 Auto Scaling policy.

For more information, see Monitoring S3 Files with Amazon CloudWatch.

CloudTrail

CloudTrail captures API calls and related events made by or on behalf of your AWS account and delivers log files to an Amazon S3 bucket that you specify. S3 Files logs management events including creating file systems, creating mount targets, and mounting file systems to compute instances. S3 Files does not log data events, such as file read and write operations.

For more information, see Logging with CloudTrail for S3 Files.

Topics

- Monitoring S3 Files with Amazon CloudWatch
- Logging with CloudTrail for S3 Files

Monitoring S3 Files with Amazon CloudWatch

You can monitor S3 Files file systems using Amazon CloudWatch, which collects and processes raw data from Amazon S3 Files into readable metrics. These metrics are retained for 15 months, so you can access historical information and gain a better perspective on how your file systems are performing.

S3 Files metric data is automatically sent to CloudWatch. Most metrics are sent at 1-minute intervals, while storage metrics are sent every 15 minutes. You can create CloudWatch alarms that send notifications when a metric exceeds a threshold you specify. You can also use CloudWatch dashboards, which are customizable home pages in the CloudWatch Console that you can use to monitor your resources in a single view. For more information, see Creating a customized CloudWatch dashboard.

S3 Files CloudWatch metrics

S3 Files metrics use the `AWS/S3/Files` namespace. All metrics are reported for a single dimension `FileSystemId`. The `AWS/S3/Files` namespace includes the following metrics:

Metric	Description	Units and valid statistics
StorageBytes	The total size of the file system in bytes, which includes data and metadata. This metric is emitted to CloudWatch every 15 minutes.	Units: Bytes. Minimum, Maximum, Average
Inodes	The total number of inodes (such as files, directories, symlinks) in an S3 Files file system. This metric is emitted to CloudWatch every 15 minutes.	Units: Count. Sum
PendingExports	The total number of files and directories pending export to the S3 bucket.	Units: Count. Sum
ImportFailures	The total number of objects that failed to import to the file system after retries (for example, incorrect IAM permissions).	Units: Count. Sum
ExportFailures	Total number of files and directories that failed export and will not be retried. This metric helps you identify terminal export failures so you can troubleshoot and take action (for example, update IAM permissions).	Units: Count. Sum
ImportAge	The age of in-progress changes that the file system imports from the linked S3 bucket. The age is measured from when each change occurred in the bucket. Use this metric to monitor whether imports are up to date. A sustained increase means imports are falling behind changes in the bucket.	Units: Seconds
ExportAge	The age of in-progress changes that the file system exports to the linked S3 bucket. The age is measured from when each change occurred in the file system. Use this metric to monitor whether exports are up to date. A sustained increase	Units: Seconds

Metric	Description	Units and valid statistics
	means exports are falling behind changes in the file system.	
`DataReadBytes`	The number of bytes read from the file system. `SampleCount` gives the number of data read operations. You can calculate data read throughput by viewing this metric per unit time.	Units: Bytes (Minimum, Maximum, Average, Sum), Count (SampleCount)
`DataWriteBytes`	The number of bytes written to the file system. `SampleCount` gives the number of data write operations. You can calculate data write throughput by viewing this metric per unit time.	Units: Bytes (Minimum, Maximum, Average, Sum), Count (SampleCount)
`MetadataReadBytes`	The number of metadata bytes read from the file system. `SampleCount` gives the number of metadata read operations.	Units: Bytes (Minimum, Maximum, Average, Sum), Count (SampleCount)
`MetadataWriteBytes`	The number of metadata bytes written to the file system. `SampleCount` gives the number of metadata write operations.	Units: Bytes (Minimum, Maximum, Average, Sum), Count (SampleCount)
`LostAndFoundFiles`	Total number of files in the lost and found directory. The lost and found directory is located in your file system's root directory under the name `.s3files-lost+found-` *`file-system-id`* . Files in the lost and found directory are not copied to your S3 bucket. When a conflict occurs due to concurrent changes to the same data in both the file system and the S3 bucket, S3 Files treats the S3 bucket as the source of truth and moves the conflicting file to a lost and found directory.	Units: Count. Sum

Metric	Description	Units and valid statistics
`ClientConnections`	The number of active client connections to a file system.	Units: Count. Sum

Client connectivity metrics

S3 Files can optimize read performance by allowing clients to read file data directly from the linked S3 bucket. To support this, the S3 Files client emits connectivity metrics that monitor whether the client can establish the necessary connections.

These metrics are emitted by the S3 Files client (amazon-efs-utils) and are published to the `efs-utils/S3Files` CloudWatch namespace. Metrics emission is enabled by default as part of the S3 Files experience.

Metric	Description	Units and valid statistics
`NFSConnectionAccessible`	Indicates whether the client can connect to the file system through the NFS mount. A value of 1 means the connection is accessible. A value of 0 means the connection is not accessible.	Units: None. Minimum, Maximum, Average
`S3BucketAccessible`	Indicates whether the client has the required permissions to read data from the linked S3 bucket. A value of 1 means the client has the necessary permissions. A value of 0 means the client does not have the necessary permissions.	Units: None. Minimum, Maximum, Average
`S3BucketReachable`	Indicates whether the linked S3 bucket and prefix exist and are reachable from the client. A value of 1 means the bucket and prefix are reachable . A value of 0 means the bucket or prefix is not reachable.	Units: None. Minimum, Maximum, Average

Accessing CloudWatch metrics

You can view S3 Files metrics using the CloudWatch console, the AWS CLI, or the CloudWatch API.

To view metrics using the CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the navigation pane, choose **Metrics**, then choose **All metrics**.
3. Choose the **S3Files** namespace.
4. Choose **File System Metrics**.
5. Select the metrics you want to view.
6. Choose the **Graphed metrics** tab to configure the graph display.

To view metrics using the AWS CLI

Use the `get-metric-statistics` command. For example, to view `DataReadBytes`:

```
aws cloudwatch get-metric-statistics \
  --namespace AWS/S3/Files \
  --metric-name DataReadBytes \
  --dimensions Name=FileSystemId,Value=file-system-id \
  --start-time 2025-01-20T00:00:00Z \
  --end-time 2025-01-20T23:59:59Z \
  --period 3600 \
  --statistics Sum
```

Logging with CloudTrail for S3 Files

Amazon S3 Files is integrated with CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in S3 Files. CloudTrail captures all API calls for S3 Files as events, including calls from the S3 Files console and code calls to the S3 Files API operations.

If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for S3 Files. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to S3 Files, the IP address from which the request was made, who made the request, when it was made, and additional details.

S3 Files information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon S3 Files, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing events with CloudTrail Event history in the *CloudTrail User Guide*.

For an ongoing record of events in your AWS account, including events for S3 Files, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs.

For more information, see the following topics in the *CloudTrail User Guide*:

- Creating a trail for your AWS account
- AWS service integrations with CloudTrail logs
- Configuring Amazon SNS notifications for CloudTrail
- Receiving CloudTrail log files from multiple Regions and Receiving CloudTrail log files from multiple accounts

All S3 Files APIs are logged by CloudTrail. For example, calls to the `CreateFileSystem`, `CreateMountTarget` and `TagResource` operations generate entries in the CloudTrail log files. S3 Files also generates CloudTrail logs when you mount your file system on a compute resource.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see CloudTrail userIdentity element in the *CloudTrail User Guide*.

S3 Files does not log data events. Data events include file read and write operations performed on the file system.

Understanding S3 Files log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so they don't appear in any specific order.

Example: CreateFileSystem

The following example shows a CloudTrail log entry that demonstrates the `CreateFileSystem` action:


```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "111122223333",
        "arn": "arn:aws:sts::111122223333:assumed-role/myRole/i-0123456789abcdef0",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "111122223333",
                "arn": "arn:aws:iam::111122223333:role/myRole",
                "accountId": "111122223333",
                "userName": "myRole"
            },
            "attributes": {
                "creationDate": "2026-03-20T12:58:28Z",
                "mfaAuthenticated": "false"
            },
            "ec2RoleDelivery": "2.0"
        }
    },
    "eventTime": "2026-03-20T17:43:19Z",
    "eventSource": "s3files.amazonaws.com",
    "eventName": "CreateFileSystem",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/2.0",
    "requestParameters": {
```

```
        "bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
        "prefix": "images/",
        "clientToken": "myClientToken",
        "roleArn": "arn:aws:iam::111122223333:role/myS3FilesRole"
    },
    "responseElements": {
        "creationTime": "Mar 20, 2026, 5:43:19 PM",
        "fileSystemArn": "arn:aws:s3files:us-west-2:111122223333:file-system/fs-
abcd123456789ef0",
        "fileSystemId": "fs-abcd123456789ef0",
        "bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
        "prefix": "images/",
        "clientToken": "myClientToken",
        "status": "creating",
        "roleArn": "arn:aws:iam::111122223333:role/myS3FilesRole",
        "ownerId": "111122223333",
        "tags": []
    },
    "requestID": "dEXAMPLE-feb4-11e6-85f0-736EXAMPLE75",
    "eventID": "eEXAMPLE-2d32-4619-bd00-657EXAMPLEe4",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::S3Files::FileSystem",
            "ARN": "arn:aws:s3files:us-west-2:111122223333:file-system/fs-
abcd123456789ef0"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "s3files.us-west-2.api.aws"
    }
}
```


Example: CreateMountTarget

The following example shows a CloudTrail log entry for the `CreateMountTarget` action:

```
{
    "eventVersion": "1.11",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "111122223333",
        "arn": "arn:aws:sts::111122223333:assumed-role/myRole/i-0123456789abcdef0",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "111122223333",
                "arn": "arn:aws:iam::111122223333:role/myRole",
                "accountId": "111122223333",
                "userName": "myRole"
            },
            "attributes": {
                "creationDate": "2026-03-20T13:09:56Z",
                "mfaAuthenticated": "false"
            },
            "ec2RoleDelivery": "2.0"
        }
    },
    "eventTime": "2026-03-20T18:05:14Z",
    "eventSource": "s3files.amazonaws.com",
    "eventName": "CreateMountTarget",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/2.0",
    "requestParameters": {
        "fileSystemId": "fs-abcd123456789ef0",
        "subnetId": "subnet-01234567890abcdef",
        "securityGroups": [
            "sg-c16d65b6"
        ]
    },
    "responseElements": {
        "availabilityZoneId": "usw2-az2",
        "ownerId": "111122223333",
        "mountTargetId": "fsmt-1234567",
        "fileSystemId": "fs-abcd123456789ef0",
        "subnetId": "subnet-01234567890abcdef",
        "ipv4Address": "192.0.2.0",
```

```
        "ipv6Address": "2001:db8::1",
        "networkInterfaceId": "eni-0123456789abcdef0",
        "vpcId": "vpc-01234567",
        "securityGroups": [
            "sg-c16d65b6"
        ],
        "status": "creating"
    },
    "requestID": "dEXAMPLE-feb4-11e6-85f0-736EXAMPLE75",
    "eventID": "eEXAMPLE-2d32-4619-bd00-657EXAMPLEe4",
    "readOnly": false,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::S3Files::FileSystem",
            "ARN": "arn:aws:s3files:us-west-2:111122223333:file-system/fs-
abcd123456789ef0"
        },
        {
            "accountId": "111122223333",
            "type": "AWS::S3Files::MountTarget",
            "ARN": "arn:aws:s3files:us-west-2:111122223333:mount-target/fsmt-1234567"
        },
        {
            "accountId": "111122223333",
            "type": "AWS::EC2::Subnet",
            "ARN": "arn:aws:ec2:us-west-2:111122223333:subnet/subnet-01234567890abcdef"
        },
        {
            "accountId": "111122223333",
            "type": "AWS::EC2::NetworkInterface",
            "ARN": "arn:aws:ec2:us-west-2:111122223333:network-interface/
eni-0123456789abcdef0"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "s3files.us-west-2.api.aws"
    }
```

```
}
```

Performance specifications

S3 Files automatically scales throughput and IOPS to match your workload without requiring you to provision or manage capacity. This page describes the performance characteristics of S3 Files.

Performance summary

Specification	Value
Aggregate read throughput per file system	Up to terabytes per second
Aggregate write throughput per file system	1–5 GiB/s
Maximum read IOPS per S3 bucket with S3 Files	No limit (attach multiple file systems to the same bucket)
Maximum write IOPS per S3 bucket with S3 Files	No limit (attach multiple file systems to the same bucket)
Maximum read IOPS per file system	250,000
Maximum write IOPS per file system	50,000
Maximum per-client read throughput	3 GiB/s

How S3 Files delivers performance

S3 Files serves data from two storage tiers, and automatically routes each operation to the tier best suited for it.

High-performance storage – The low-latency storage layer within your file system where actively used file data and metadata reside. S3 Files automatically manages this storage, copying data onto it when you access files and removing data that has not been read within a configurable expiration window. You pay a storage rate for data residing on the high-performance storage.

Direct from S3 – S3 Files streams file reads directly from your S3 bucket in two cases: when the file's data is not stored in the file system's high-performance storage, and for large reads >= 1 MiB,

even when the data also resides on the file system's high-performance storage. The S3 bucket is optimized for high throughput while the file system's high-performance storage layer is optimized for low-latency access. This tiering approach for streaming data directly from S3 bucket provides high throughput for sequential reads, making it well suited for analytics, media processing, and other streaming workloads. S3 Files asynchronously imports data for small files (< 128 KiB by default) to the file system's high-performance storage for low latency access on subsequent reads.

Since S3 Files automatically applies this two-tier model, you do not have to choose between latency and throughput. Small-file workloads get file system performance. Large-file workloads get S3 throughput. Mixed workloads get both.

Read performance

Read throughput scales with the number of connected compute instances and the degree of parallelism within each instance. The maximum per-client read throughput is 3 GiB/s. S3 Files supports up to terabytes per second of aggregate read throughput and up to 250,000 read IOPS per file system.

Write performance

Writes go to the high-performance storage and are durable immediately. Depending on the region, S3 Files supports 1–5 GiB/s of aggregate write throughput and up to 50,000 write IOPS per file system. Write performance scales elastically with workload activity.

When you modify a file in the file system, S3 Files waits approximately 60 seconds, aggregating any successive changes to the file in that time, before copying to your S3 bucket. This means that rapid successive writes to the same file are captured in a single S3 PUT request rather than generating a new object version for every individual change, reducing your S3 request costs and storage costs. If you continue to modify the file after S3 Files has copied your changes back to the S3 bucket, it will copy subsequent changes as needed.

First access latency

The first time you access a directory, S3 Files imports metadata for all files in that directory and, depending on your import configuration, data for small files. So your initial access takes longer than subsequent operations. Once imported, all subsequent directory listings and file access return at low latency.

Synchronization performance

S3 Files synchronizes changes between your file system and S3 bucket in the background.

Importing changes from S3 – When another application adds or modifies an object in your S3 bucket, S3 Files reflects the change in your file system typically within seconds. S3 Files processes up to 2,400 object changes per second per file system, with import data throughput of up to 700 megabytes per second.

Exporting changes to S3 – When you write a file through the file system, S3 Files batches your changes for approximately 60 seconds to consolidate rapid successive writes into a single S3 object version, reducing your S3 request and storage version costs. After the batching window, S3 Files copies the file to your S3 bucket. S3 Files exports up to 800 files per second per file system, with export data throughput of up to 2,700 megabytes per second.

Operation metric	Value	Unit
Import from S3 bucket IOPS	2,400	objects per second per file system
Import from S3 bucket throughput	700	megabytes per second
Export to S3 bucket IOPS	800	files per second per file system
Export to S3 bucket throughput	2,700	megabytes per second

Amazon S3 uses a flat storage structure where objects are identified by their key names. While S3 Files lets you organize your data in directories, S3 has no native concept of directories. What appears as a directory in your file system is a common prefix shared by the keys of the objects within the S3 bucket. Additionally, S3 objects are immutable and do not support atomic renames. As a result, when you rename or move a file, S3 Files must write the data to a new object with the updated key and delete the original. When you rename or move a directory, S3 Files must repeat this process for every object that shares that prefix. Therefore, when you rename or move a directory containing tens of millions of files, your S3 request costs and the synchronization time increase significantly. A directory rename of 100,000 files takes a few minutes to fully reflect in the S3 bucket, though the rename is instant on the file system. For more information, see Understanding the impact of rename and move operations.

If your workload generates changes faster than the synchronization rate, S3 Files queues the changes and processes them in order. You can monitor the count of pending exports using the `PendingExports` CloudWatch metric. For more information, see Monitoring S3 Files with Amazon CloudWatch.

Monitoring performance

You can monitor your file system's performance using Amazon CloudWatch. S3 Files publishes metrics including `DataReadBytes`, `DataWriteBytes`, `MetadataReadBytes`, and `MetadataWriteBytes`, which you can use to track throughput and IOPS over time. For more information, see Monitoring S3 Files with Amazon CloudWatch.

Security for S3 Files

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

Security of the cloud

AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS Compliance Programs. To learn about the compliance programs that apply to Amazon S3 Files, see AWS Services in Scope by Compliance Program.

Security in the cloud

Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon S3 Files.

Data Protection

The AWS shared responsibility model applies to data protection in S3 Files. As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data Privacy FAQ. For information about data protection in Europe, see the AWS Shared Responsibility Model and GDPR blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail. For information about using CloudTrail trails to capture AWS activities, see Working with CloudTrail trails in the *AWS CloudTrail User Guide*.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-3 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-3.

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with S3 Files or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Topics

- Encryption

- How S3 Files works with IAM
- Control client access to an S3 Files file system
- POSIX permissions in S3 Files

Encryption

S3 Files provides comprehensive encryption capabilities to protect your data both at rest and in transit.

Encryption at rest

Your S3 bucket is encrypted using Amazon S3's encryption mechanisms. For information on encryption of data in S3, see Protecting data with encryption.

S3 Files encrypts data at rest in the S3 file system using server-side encryption. Server-side encryption is the encryption of data at its destination by the application or service that receives it. In S3 file systems, data and metadata are encrypted by default before being written to storage and are automatically decrypted when read. These processes are handled transparently by S3 Files, so you don't need to modify your applications. All data at rest in the file system is encrypted using one of the following methods:

- (Default) Server-side encryption with Amazon S3-managed encryption keys (SSE-S3)
- Server-side encryption with AWS Key Management Service (SSE-KMS)

There are additional charges for using AWS KMS keys. For more information, see AWS KMS key concepts in the *AWS Key Management Service Developer Guide* and AWS KMS pricing.

Server-side encryption with Amazon S3-managed encryption keys (SSE-S3)

This is the default method for encrypting data at rest in your S3 file system. Amazon S3-managed encryption keys are a collection of keys that Amazon S3 owns and manages. S3 Files owns and manages encryption of your data and metadata at rest in your S3 file system when you use an Amazon S3-managed encryption key. For more information about Amazon S3-managed encryption keys, see Protecting data with Amazon S3 managed keys.

Server-side encryption with AWS Key Management Service (SSE-KMS)

While creating your file system, you can choose to configure an AWS Key Management Service (AWS KMS) key that you manage. When you use SSE-KMS encryption with an S3 file system the AWS KMS keys must be in the same Region as the file system.

S3 Files key policies for AWS KMS

Key policies are the primary way to control access to customer managed keys. For more information on key policies, see Key policies in AWS KMS in the *AWS Key Management Service Developer Guide*. The following list describes all the AWS KMS–related permissions that are supported by S3 Files for encrypting file systems at rest:

kms:Encrypt

(Optional) Encrypts plaintext into ciphertext. This permission is included in the default key policy.

kms:Decrypt

(Required) Decrypts ciphertext. Ciphertext is plaintext that has been previously encrypted. This permission is included in the default key policy.

kms:ReEncrypt

(Optional) Encrypts data on the server side with a new customer managed key, without exposing the plaintext of the data on the client side. The data is first decrypted and then re-encrypted. This permission is included in the default key policy.

kms:GenerateDataKeyWithoutPlaintext

(Required) Returns a data encryption key encrypted under a customer managed key. This permission is included in the default key policy under kms:GenerateDataKey*.

kms:CreateGrant

(Required) Adds a grant to a key to specify who can use the key and under what conditions. Grants are alternate permission mechanisms to key policies. For more information on grants, see Grants in AWS KMS in the *AWS Key Management Service Developer Guide*. This permission is included in the default key policy.

kms:DescribeKey

(Required) Provides detailed information about the specified customer managed key. This permission is included in the default key policy.

kms:ListAliases

(Optional) Lists all of the key aliases in the account. When you use the console to create an encrypted file system, this permission populates the Select KMS key list. We recommend using this permission to provide the best user experience. This permission is included in the default key policy.

Key states and their effects

The state of your KMS key directly affects access to your encrypted file system:

Enabled

Normal operation - full read and write access to the file system.

Disabled

File system becomes inaccessible after some time. Can be re-enabled.

Pending deletion

File system becomes inaccessible. Deletion can be canceled during the waiting period. Note that after cancelling key deletion, the key needs to be moved to enabled state.

Deleted

File system permanently inaccessible. This action cannot be reversed.

> **Warning**
>
> If you disable or delete the KMS key used for your file system, or revoke S3 Files access to the key, your file system will become inaccessible. This can result in data loss if you don't have backups. Always ensure you have proper backup procedures in place before making changes to encryption keys.

Encryption in transit

S3 Files requires encryption of data in transit using Transport Layer Security (TLS). When you mount your file system using the mount helper, all data traveling between your client and the file system is encrypted using TLS. The mount helper initializes efs-proxy process to establish a secure TLS connection with your file system. The mount helper also creates a process called amazon-

efs-mount-watchdog that monitors the health of mounts, and is started automatically the first time an S3 file system is mounted. It ensures that each mount's efs-proxy process is running, and stops the process when the file system is unmounted. If for some reason the process is terminated unexpectedly, the watchdog process restarts it.

The following describes how TLS encryption in transit works:

1. A secure TLS connection is established between your client and the file system
2. All NFS traffic is routed through this encrypted connection
3. Data is encrypted before transmission and decrypted upon receipt

Encryption of data in transit changes your NFS client setup. When you inspect your actively mounted file systems, you see one mounted to 127.0.0.1, or localhost, as in the following example.

```
$ mount | column -t
127.0.0.1:/  on  /home/ec2-user/s3files        type  nfs4
 (rw,relatime,vers=4.1,rsize=1048576,wsize=1048576,namlen=255,hard,proto=tcp,port=20127,timeo=6
```

You mount your file system using the mount helper, which always encrypts data in transit using TLS. Therefore, while mounting, your NFS client is reconfigured to mount to a local port.

How S3 Files works with IAM

This page describes how AWS Identity and Access Management (IAM) works with S3 Files and how you can use IAM policies to control access to your file systems.

S3 Files uses IAM for two distinct types of access control:

- **API access** — Controls who can create, manage, and delete S3 Files resources such as file systems, mount targets, and access points. You control this access using identity-based policies attached to IAM users, groups, or roles.
- **Client access** — Controls what clients (your mounted compute resources) can do with the file system once they connect, such as reading, writing, or accessing files as the root user. You control this access using a combination of resource-based policies, identity-based policies, access points, and POSIX permissions.

Using IAM, you can permit clients to perform specific actions on a file system, including read-only, write, and root access. An "allow" permission on an action in either an IAM identity policy or a file

system resource policy allows access for that action. The permission does not need to be granted in both an identity and a resource policy.

Your S3 bucket policies on your linked S3 bucket also govern access from your compute resource and your file system to your S3 bucket. You must also make sure that the bucket policies of your source bucket don't deny access from your compute resource or file system. For more details, see Bucket policies for Amazon S3.

Identity-based policies

Identity-based policies are JSON policies that you attach to IAM users, groups, or roles. You can provide these permissions by writing custom policies or by attaching an AWS managed policy. For more information about available managed policies for both API access and client access, see AWS managed policies for Amazon S3 Files.

S3 Files also optimizes read performance by allowing clients to read file data directly from the source S3 bucket. When you mount an S3 file system on your compute resource, you must add an inline policy to the IAM role of your compute resource which grants permissions to read objects from the specified S3 bucket. The mount helper uses these permissions to read the S3 data. For more details on this policy, see IAM role for attaching your file system to AWS compute resources.

Resource-based policies

A file system policy is an IAM resource-based policy that you attach directly to a file system to control client access. You can use file system policies to grant or deny permissions for clients to perform operations such as mounting, writing, and root access.

A file system either has an empty (default) file system policy or exactly one explicit policy. S3 file system policies have a 20,000 character limit. For information on creating and managing file system policies, see Creating file system policies.

S3 Files actions for clients

You can specify the following actions in a file system policy to control client access:

Action	Description
`s3files:ClientMount`	Provides read-only access to a file system.
`s3files:ClientWrite`	Provides write permissions on a file system.

Action	Description
`s3files:ClientRootAccess`	Provides use of the root user when accessing a file system.

S3 Files condition keys for clients

You can use the following condition keys in the `Condition` element of a file system policy to further refine access control:

Condition key	Description	Operator
`s3files:AccessPointArn`	ARN of the S3 Files access point to which the client is connecting.	String

POSIX permissions

After IAM authorization succeeds, S3 Files enforces standard POSIX (Unix-style) permissions at the file and directory level. POSIX permissions control access based on the user ID (UID), group ID (GID), and permission bits (read, write, execute) associated with each file and directory. Access points can enforce a specific POSIX user identity for all requests, simplifying access management for shared datasets. For more information, see Creating access points for an S3 file system.

Security groups

Security groups act as a network-level firewall that controls traffic between your compute resources and the file system's mount targets. For details on configuring security groups to get started on S3 Files, see Security groups.

AWS managed policies for Amazon S3 Files

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you

reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AmazonS3FilesFullAccess

You can attach the `AmazonS3FilesFullAccess` policy to your IAM identities. This policy grants full access to Amazon S3 Files, including permissions to create and manage file systems, mount targets, and access points. For more information about this policy, see AmazonS3FilesFullAccess in the AWS Managed Policy Reference.

AWS managed policy: AmazonS3FilesReadOnlyAccess

You can attach the `AmazonS3FilesReadOnlyAccess` policy to your IAM identities. This policy grants read-only access to Amazon S3 Files, including permissions to view file systems, mount targets, access points, and related configurations. For more information about this policy, see AmazonS3FilesReadOnlyAccess in the AWS Managed Policy Reference.

AWS managed policy: AmazonS3FilesClientFullAccess

You can attach the `AmazonS3FilesClientFullAccess` policy to your IAM identities. This policy grants full client access to S3 Files file systems, including the ability to mount, read, write, and access files as the root user. For more information about this policy, see AmazonS3FilesClientFullAccess in the AWS Managed Policy Reference.

AWS managed policy: AmazonS3FilesClientReadWriteAccess

You can attach the `AmazonS3FilesClientReadWriteAccess` policy to your IAM identities. This policy grants read and write client access to S3 Files file systems, including the ability to mount, read, and write. This policy does not grant root access. For more information about this policy, see AmazonS3FilesClientReadWriteAccess in the AWS Managed Policy Reference.

AWS managed policy: AmazonS3FilesClientReadOnlyAccess

You can attach the `AmazonS3FilesClientReadOnlyAccess` policy to your IAM identities. This policy grants read-only client access to S3 Files file systems, including the ability to mount and read from the file system. For more information about this policy, see AmazonS3FilesClientReadOnlyAccess in the AWS Managed Policy Reference.

AWS managed policy: AmazonS3FilesCSIDriverPolicy

You can attach the `AmazonS3FilesCSIDriverPolicy` policy to your IAM identities. This policy grants permissions for the Amazon EFS Container Storage Interface (CSI) driver to manage S3 Files access points on behalf of Amazon EKS clusters. For more information about this policy, see AmazonS3FilesCSIDriverPolicy in the AWS Managed Policy Reference.

AWS managed policy: AmazonElasticFileSystemsUtils

You can attach the `AmazonElasticFileSystemsUtils` policy to your IAM identities. This policy grants permissions for the S3 Files client utilities (amazon-efs-utils) to perform operations such as describing mount targets, publishing CloudWatch metrics and logs, and communicating with AWS Systems Manager. For more information about this policy, see AmazonElasticFileSystemsUtils in the AWS Managed Policy Reference.

Amazon S3 Files updates to AWS managed policies

View details about updates to AWS managed policies for Amazon S3 Files since S3 Files began tracking these changes.

Change	Description	Date
`AmazonElasticFileSystemsUtils` — Updated	Added Amazon CloudWatch PutMetricData permissions to support publishing client connectivity metrics.	April 7, 2026
`AmazonS3FilesCSIDriverPolicy` — Added	New managed policy that grants permissions for the Amazon EFS CSI driver to manage S3 Files access points on behalf of Amazon EKS clusters.	April 7, 2026

Change	Description	Date
`AmazonS3FilesClientReadOnlyAccess` — Added	New managed policy that grants read-only client access to S3 Files file systems.	April 7, 2026
`AmazonS3FilesClientReadWriteAccess` — Added	New managed policy that grants read and write client access to S3 Files file systems.	April 7, 2026
`AmazonS3FilesClientFullAccess` — Added	New managed policy that grants full client access to S3 Files file systems, including root access.	April 7, 2026
`AmazonS3FilesReadOnlyAccess` — Added	New managed policy that grants read-only access to S3 Files resources.	April 7, 2026
`AmazonS3FilesFullAccess` — Added	New managed policy that grants full access to S3 Files resources.	April 7, 2026
S3 Files started tracking changes	Amazon S3 Files started tracking changes for its AWS managed policies.	April 7, 2026

Control client access to an S3 Files file system

This topic describes how to use file system policies, IAM identity policies, and network controls to control which clients can access your S3 Files file system and through which access points.

S3 Files access points

S3 Files access points are application-specific entry points to a file system that simplify managing data access at scale. You can use access points to enforce user identities and permissions for all file system requests made through the access point, and restrict clients to only access data within a specified root directory and its subdirectories.

Access points scope what a client can do on the file system. They do not scope which clients can use them. Any role in your account with `s3files:ClientMount` permission can mount through any access point on the file system unless a file system policy restricts it. To control which clients can use which access point, combine three layers: network controls (prevent the connection), a file system resource policy (explicit deny that survives identity policy changes), and an IAM identity policy (least privilege on the role).

S3 Files evaluates `s3files:ClientMount`, `s3files:ClientWrite`, and `s3files:ClientRootAccess` against the file system policy on every mount. The `s3files:AccessPointArn` condition key is also evaluated on every mount, so you can write a single deny statement that blocks access through any access point except the one you specify. If any required action is denied or not granted, the mount fails before any file operations occur. The following two sections cover the two common requirements: restricting a workload to a specific access point, and denying a workload access to the file system altogether.

Restrict access to a specific access point

This pattern applies when a role (for example, an EC2 instance role or an ECS task role) must mount the file system through one access point only. The condition key is `s3files:AccessPointArn`. The policy requires both an `Allow` for the intended access point and an explicit `Deny` for every other access point. IAM allow grants are additive across identity and resource policies; without an explicit deny, a separate policy can grant access to a different access point.

A role is allowed to mount and write only when the request targets `fsap-ID`, and is explicitly denied on every other access point on this file system. An explicit deny in a resource policy cannot be overridden by identity policies, so this restriction holds even if a broader IAM policy is attached to the role later.

Mount command:

```
sudo mount -t s3files -o accesspoint=fsap-ID fs-ID:/ /mnt/s3files
```

Deny a workload access through any access point

This pattern applies when a role shares an account with an S3 Files file system but must not mount it through any access point or directly. To block all access point usage, apply three layers ordered from strongest to weakest isolation guarantee.

1. Network controls. Remove the security group rule that grants `compute-b-role`'s instances access to the mount target on TCP port 2049. Network controls are the most resilient layer because they prevent the mount attempt before any credential exchange occurs. Even if a misconfigured policy grants mount permissions, the NFS connection cannot reach the mount target.

2. Explicit deny in the file system policy. A deny statement in the file system policy rejects the mount even if the role is granted broader IAM permissions later. A file system resource policy provides a deny that persists even if someone with `iam:PutRolePolicy` modifies the role's identity policies. Only principals with `s3files:PutFileSystemPolicy` can change this layer.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyComputeBOnFileSystem",
      "Effect": "Deny",
      "Principal": { "AWS": "arn:aws:iam::ACCOUNT:role/compute-b-role" },
      "Action": "s3files:Client*",
      "Resource": "*"
    }
  ]
}
```

3. IAM identity policy. Do not grant `s3files:Client*` permissions to the role. This is the final layer and the easiest to misconfigure, since any administrator with `iam:PutRolePolicy` can grant mount permissions to a role.

Without a file system policy, any role can mount through any access point

A file system resource policy provides a deny that cannot be removed by someone who only has permission to modify identity policies. Without a file system policy, any role with `s3files:ClientMount` in its identity policy can mount through any access point on the file system. We recommend that you attach a file system policy to every file system with a deny baseline, then add allow statements per workload.

Best practices for access point isolation

The following table summarizes recommended approaches for common access point isolation goals.

Goal	Recommended approach
Restrict a workload to a single access point	Allow the role on the specific access point ARN, plus an explicit deny on every other access point. See "Restrict access to a specific access point."
Block a role in the same account from mounting the file system	Remove network access to port 2049, deny in file system policy, and do not grant in IAM.
Multiple workloads on the same file system, each scoped to their own access point	Maintain one allow + deny pair per role in the file system policy, each scoped to a different access point ARN.
Cross-account access through an access point	Use the file system policy to grant the cross-account role on the specific access point ARN. Do not rely on identity policies alone.

POSIX permissions in S3 Files

POSIX file systems require ownership and permissions on every file. Amazon S3 stores data as objects with access controlled by IAM policies and bucket policies and does not natively store file ownership or permission attributes. S3 Files bridges this difference by storing POSIX metadata, including the owner, group, and file permissions, as user-defined metadata on each S3 object. When you create or modify a file through S3 Files, the service writes all permission metadata automatically. When an object is written to your S3 bucket outside the file system, for example through the Amazon S3 console, AWS CLI, or AWS SDK, no POSIX metadata exists on that object. S3 Files assigns those objects a default value of `root:root` ownership and `644` permissions, which allow non-root users to read the file but not write to it. If these defaults meet your access requirements, no action is required. If your application requires it, you can override the default permissions on those objects. This tutorial walks through applying custom permissions.

POSIX permissions and direct S3 access

S3 Files applies access controls on two independent paths. POSIX user and group permissions govern the operating system users and processes that access the file-system mount. IAM and bucket policies govern the Amazon S3 API.

POSIX permissions do not restrict access made directly through the Amazon S3 API. Any principal with `s3:GetObject` on the bucket can read object contents directly, regardless of the POSIX permissions stored on the object. Treat IAM and bucket policies as the control for direct S3 access, and grant them following the principle of least privilege.

How S3 Files stores POSIX metadata

S3 Files stores POSIX information as user-defined metadata on each object. The following metadata keys control ownership and permissions:

Metadata key	Description	Example value
`x-amz-meta-file-owner`	POSIX user ID (UID)	`1000`
`x-amz-meta-file-group`	POSIX group ID (GID)	`1000`
`x-amz-meta-file-permissions`	File type and permission bits (octal)	`0100644`

When an object has no POSIX metadata, S3 Files applies the following defaults:

- **Files:** `root:root` ownership with `0644` permissions (`-rw-r--r--`)
- **Directories:** `root:root` ownership with `0755` permissions (`drwxr-xr-x`)

Under these defaults, all users can read files and traverse directories, but only the owner (root) can write. Files are not executable by default.

Setting POSIX metadata on existing objects

Use the following procedure to apply POSIX ownership and permissions to objects that already exist in your bucket.

Step 1: Identify the target user's UID and GID

For this example, the target user is `ec2-user` with UID 1000:

```
$ id ec2-user
```

```
uid=1000(ec2-user) gid=1000(ec2-user) groups=1000(ec2-user)
```

Step 2: Update object metadata using copy-object

The `aws s3api copy-object` command copies the object in place with updated metadata. Replace *BUCKET_NAME* and *OBJECT_KEY* with your values.

The `--metadata-directive REPLACE` option replaces all user-defined metadata on the object. Include `--content-type` to preserve the object's content type, because REPLACE resets it to the default (`binary/octet-stream`). If you use other metadata fields, include them in the `--metadata` parameter.

```
aws s3api copy-object \
    --bucket BUCKET_NAME \
    --key OBJECT_KEY \
    --copy-source BUCKET_NAME/OBJECT_KEY \
    --metadata-directive REPLACE \
    --content-type CONTENT_TYPE \
    --metadata '{"file-owner":"1000", "file-group":"1000", "file-
permissions":"0100644"}'
```

Step 3: Verify metadata was applied

Run the following command to confirm the metadata:

```
aws s3api head-object --bucket BUCKET_NAME --key OBJECT_KEY
```

The response includes your POSIX metadata:

```
{
    "AcceptRanges": "bytes",
    "LastModified": "Mon, 18 May 2026 14:43:00 GMT",
    "ContentLength": 7214,
    "ETag": "\"abc123...\"",
    "ContentType": "image/png",
    "Metadata": {
        "file-owner": "1000",
        "file-group": "1000",
        "file-permissions": "0100644"
    }
}
```

Step 4: Verify permissions on the S3 file system

```
$ ls -ltr /mnt/s3files/
-rw-r--r--. 1 ec2-user ec2-user 7214 May 18 14:43 s3.png
```

The file is now owned by `ec2-user` and no longer requires root to access.

Setting POSIX metadata on a directory

A directory's ownership and permissions come from a single object: the zero-byte directory marker at the prefix key with a trailing slash, for example `data/`. S3 Files reads POSIX metadata from that one marker object to determine the directory's owner, group, and permissions. The objects inside the directory do not affect it. Each child object carries its own metadata and governs only itself.

When a directory's objects were written to your bucket before the file system existed, the directory marker has no POSIX metadata. This also applies when the objects were written through the Amazon S3 console, AWS CLI, or AWS SDK. In that case, S3 Files assigns the directory the default `root:root` ownership with `0755` permissions. With these permissions, all users can read and traverse the directory, but only the owner (root) can create or delete files within it. As a result, a non-root user receives a permission error when writing to the directory.

For directories, set the `file-permissions` value with directory bits, for example `0040755`. For more values, see Permission values reference.

To update the directory's metadata in place, copy the directory marker object onto itself with new metadata. This keeps any content stored in the object.

```
aws s3api copy-object \
    --bucket BUCKET_NAME \
    --key data/ \
    --copy-source BUCKET_NAME/data/ \
    --metadata-directive REPLACE \
    --metadata '{"file-owner":"1001", "file-group":"1001", "file-
permissions":"0040755"}'
```

If the command returns a 404 error, the directory marker object doesn't exist. The directory is implied only by its child keys, so it uses the `root:root` default. Create the marker object with the metadata.

```
aws s3api put-object \
    --bucket BUCKET_NAME \
```

```
    --key data/ \
    --metadata '{"file-owner":"1001", "file-group":"1001", "file-
permissions":"0040755"}'
```

Setting metadata on the directory marker changes only the directory itself. To change a child file's ownership, set metadata on that file object. For instructions, see Setting POSIX metadata on existing objects.

Setting POSIX metadata on new uploads

You can include POSIX metadata when you upload objects to your bucket.

Using aws s3 cp

```
aws s3 cp localfile.txt s3://BUCKET_NAME/localfile.txt \
    --metadata '{"file-owner":"1000", "file-group":"1000", "file-
permissions":"0100644"}'
```

Using aws s3api put-object

```
aws s3api put-object \
    --bucket BUCKET_NAME \
    --key localfile.txt \
    --body localfile.txt \
    --metadata '{"file-owner":"1000", "file-group":"1000", "file-
permissions":"0100644"}'
```

Verify metadata on the uploaded object

```
aws s3api head-object --bucket BUCKET_NAME --key localfile.txt
```

Permission values reference

The following table lists common POSIX permission values and their use cases.

Octal value	File mode	Use case
0100644	-rw-r--r--	Standard file: owner read/write, others read-only
0100664	-rw-rw-r--	Shared file: owner and group read/write, others read-only

Octal value	File mode	Use case
0100755	-rwxr-xr-x	Executable: owner full access, others read and execute
0100666	-rw-rw-rw-	World-writable file: all users can read and write
0040755	drwxr-xr-x	Standard directory: owner full access, others read and traverse

Best practices

Upload files through the S3 file system whenever possible. S3 Files automatically stores the file's ownership and permissions as object metadata, so no additional steps are required. When uploading directly to the S3 bucket, include POSIX metadata in the upload command to ensure the correct ownership and permissions are set from the start.

How S3 Files is metered

S3 Files is a shared file system linked to your S3 bucket, designed to deliver low-latency file access while keeping costs proportional to your active working set. The file system maintains a view of the objects in your bucket and intelligently translates your file system operations into efficient S3 requests on your behalf. As you work with specific files and directories through the file system, associated file metadata and contents are placed onto the file system's high-performance storage, in particular the portions that benefit from low-latency access. Many read operations bypass the file system entirely, with data served directly from your S3 bucket at S3 GET request rates with no S3 file data charges. Your authoritative data always remains in your S3 bucket. When you write data, it is stored on the file system's highly durable high-performance storage and then synced back to your S3 bucket, keeping the file system and your S3 bucket consistent in both directions.

With S3 Files, you pay a storage charge for the fraction of active data on the file system's high-performance storage, and you pay data access charges when reading from and writing to the file system's high-performance storage. This page explains how each dimension is metered so you can understand and optimize your costs. For pricing by AWS Region, see Amazon S3 Pricing.

How file system storage is metered

When you access data, the file system loads portions of file metadata and contents on demand onto the file system's high-performance storage, delivering fast reads without duplicating your

entire dataset. You configure a file size threshold (default 128 KiB) that determines which files are stored on high-performance storage. Files at or below this threshold benefit most from low-latency access. Files that exceed the threshold are streamed directly from your S3 bucket and incur no S3 Files storage charges. Data not accessed within a configurable window (1 – 365 days, default 30 days) automatically expires from high-performance storage. You pay a storage rate for the fraction of active data residing on high-performance storage. Typically, the fraction is small, since large files stream directly from your S3 bucket, stale data expires automatically, and only small, latency-sensitive files are stored on high-performance storage. The minimum billable file size on high-performance storage is 10 KiB.

How data access is metered

You pay data access charges for metadata operations and for reads and writes to the file system's high-performance storage. Large file reads (1 MiB or larger file IO) are always streamed directly from your S3 bucket, even if data resides on the file system's high-performance storage. S3 is optimized for high throughput reads, while high-performance storage is optimized for low-latency small-file access. Direct reads incur S3 GET requests and an S3 Files metadata read (4 KiB) with no file read charges. Background sync operations also incur data access charges and S3 request charges. Importing data onto high-performance storage incurs write charges, and exporting changes back to your S3 bucket incurs read charges.

How data access is metered from the file system

S3 Files meters every file system operation as either a read or a write and applies to a file or metadata. Each operation has a minimum metered size and is then rounded up to the next 1 KiB increment. This means every operation falls into one or two of four categories: a data read, a metadata read, a data write, or a metadata write. For example, reading a file is metered as both a data read and a metadata read, while renaming a file is metered as a metadata read and a metadata write. No single operation is ever metered as more than two categories.

File reads from high-performance storage are metered at the size of the data read, with a minimum of 32 KiB per read operation.

File writes to high-performance storage are metered at the size of the data written, with a minimum of 32 KiB per write operation.

Metadata reads are metered at a 4 KiB minimum size and apply as S3 Files reads. Metadata read operation examples include listing a directory and viewing file attributes.

Metadata writes are metered at a 4 KiB minimum size and apply as S3 Files writes. Metadata write operation examples include creating or deleting files and directories, renaming, changing permissions, and calling `fsync`.

How streaming directly from your S3 bucket is metered

S3 Files streams reads directly from your S3 bucket in two cases: the file's data is not stored on high-performance storage, or the read is 1 MiB or larger, even if the data also resides on high-performance storage. This design reflects the strengths of each storage layer. The S3 bucket is optimized for high throughput, while the file system is optimized for low-latency access.

For small files (less than 128 KiB by default), S3 Files asynchronously imports data onto high-performance storage so that subsequent reads are served at low latency. For direct bucket streams, you pay for the S3 GET requests and an S3 Files metadata read (4 KiB), with no file read charges.

How bucket synchronization is metered

S3 Files automatically keeps your file system and your linked S3 bucket synchronized. Synchronization is metered as file reads, file writes, and S3 request charges. For more information, see Understanding how synchronization works.

Importing data onto the file system: When S3 copies data from your S3 bucket onto high-performance storage based on your settings, the operation is metered as a file system write. Import writes occur when you access a directory for the first time, when you read a file not stored on high-performance storage, and when S3 Files reflects changes made directly to your S3 bucket. The metered size is the amount of data written to high-performance storage and a metadata write.

Exporting changes to your S3 bucket: When S3 Files copies your file system changes back to your S3 bucket, the operation is metered as a file system metadata and a file read. Only data read from high-performance storage counts toward this charge. For example, if you append data to a file, S3 Files uses `UploadPartCopy` to avoid importing the entire object onto high-performance storage before appending. This optimizes your high-performance storage costs.

Rename and move operations: S3 buckets do not natively support directories or renames. What appears as a directory in your S3 file system is a common prefix shared by object keys in the bucket, and S3 objects are immutable. As a result, when you rename or move a file, S3 Files copies the data to a new object with the updated key (metered as an S3 PUT request) and deletes the original. The synchronization is metered as a metadata read and a file read based on the data's location. If the file data is not stored on high-performance storage, only a 4 KiB metadata read applies. For file renames or directory moves, S3 Files repeats this copy-and-delete for every object

under that prefix. For more information, see Understanding the impact of rename and move operations.

File data expiration: File data not accessed within a configurable window of 1 to 365 days (default 30 days) automatically expires from high-performance storage. Expiration incurs no data access or metadata charges.

Metadata updates: Your file system metadata (inodes) reflects the contents of your linked S3 bucket. As your bucket changes, metadata is updated to stay consistent with the current state of your bucket. Metadata for accessed directories never expires. You can use the Inodes CloudWatch metric to monitor your metadata usage. Metadata expiration incurs no charges. POSIX metadata changes such as chown and chmod update the object in your S3 bucket. S3 objects are immutable, so S3 Files writes a new version of the object with the updated metadata, metered as an S3 PUT request and S3 Files export. Each metadata change creates an additional version of the object.

Metering examples

Listing a large directory for the first time

When you first list a directory, S3 Files imports metadata for all files in that directory. Each file's metadata import is metered as a 4 KiB write. Depending on your import configuration (default 128 KiB), S3 Files also prefetches and copies data for small files in that directory onto the file system's high-performance storage to optimize for the lowest latency. Each file's data import is metered as a write at the file's size (32 KiB minimum). You can control which files have their data imported by configuring your import rules. For more information, see Customizing synchronization for S3 Files.

Reading a small file not stored on high-performance storage

S3 Files streams read directly from your S3 bucket to your client and asynchronously imports data to the file system's high-performance storage, so future reads are faster. This is metered as a file system read at the size of the data transferred (32 KiB minimum). The asynchronous import of data into the file system's high-performance storage is metered as a write at the size of the data transferred. A similar process is followed when you read a file whose data has been expired from the file system. When files expire from the high-performance storage, it does not incur any file system operation charges.

Writing to the file system

All file writes are stored on high-performance storage and metered at the size of the data written with a 32 KiB minimum. S3 Files waits for a period of inactive write activity (60 seconds) in order

to aggregate successive changes to the same file before copying to your S3 bucket. Rapid writes are captured in a single S3 PUT rather than generating a new object version for each individual change. This reduces both S3 request costs and file high-performance storage costs. This bucket synchronization is metered as a file system read for data read from high-performance storage, and an S3 PUT request.

S3 Files best practices

This page describes the recommended best practices for working with S3 file systems.

Performance and cost optimization

- **Parallelize your workloads** – S3 Files is designed to support highly parallel workloads. Distributing reads across multiple files and multiple compute instances helps maximize aggregate throughput. You can also create multiple file systems scoped to different specific prefixes within the same bucket (instead of creating one file system over the entire bucket) to scale horizontally and improve aggregate throughput.
- **Scope your file system to the smallest prefix your workload needs to minimize impact of renames** – S3 has no native concept of directories. When you rename or move a directory, S3 Files must write the data to a new object with the updated key and delete the original for every file in that directory. Renaming directories with tens of millions of files can significantly increase S3 request costs and synchronization time. Scope your file system to your active dataset, or structure your data so that directories you expect to rename contain fewer files. For more information, see Understanding the impact of rename and move operations.
- **Use large IO sizes** – S3 Files meters each read and write operation at a minimum of 32 KB. Using larger IO sizes (1 MB or more) amortizes per-operation overhead and is more cost effective than many small reads or writes. When using the mount helper, the default NFS read and write buffer sizes are set to 1 MB for optimal performance.
- **Tune your sizeLessThan value in import configuration to match your file sizes** – By default, S3 Files caches data for files smaller than 128 KB when you first access a directory. Files larger than this threshold are read directly from S3. If your workload performs small, latency-sensitive reads on larger files, increase the sizeLessThan threshold to match the file sizes you need on the file system's high performance storage for low-latency access. However, a higher threshold also increases the amount of data that S3 Files imports onto high-performance storage, which affects your metered usage. For more information, see Customizing synchronization for S3 Files. For S3 Files pricing, see Amazon S3 pricing on the AWS website.

- **Set expiration windows to match your workload lifecycle** – Data that has not been read within the expiration window is automatically removed from the file system. For short-lived workloads such as batch jobs or training runs, use a shorter expiration (1–7 days) to minimize storage costs. For workloads that revisit the same data over weeks, use a longer expiration (30–90 days) to continue benefiting from the low latency. For more information, see Customizing synchronization for S3 Files.
- **Use prefix-scoped rules for mixed workloads** – If your bucket contains both frequently accessed and infrequently accessed data, create separate import rules for each prefix. This lets you import data aggressively for hot prefixes while keeping cold prefixes metadata-only. For more information, see Customizing synchronization for S3 Files.
- **Create a mount target in every Availability Zone** – We recommend creating one mount target in each Availability Zone you operate in so that you can reduce cross-AZ data transfer costs and improve performance. This ensures that your compute resources always have a local network path to the file system, improving both availability and latency. When you create a file system using the AWS Management Console, S3 Files automatically creates one mount target in every Availability Zone in your selected VPC.

Synchronization

- **Understand the S3 Files consistency model** – When a file in the file system is modified at the same time as its corresponding object in the S3 bucket, S3 Files treats the S3 bucket as the source of truth and moves the file to the lost and found directory. To avoid conflicts, designate one path (file system or S3) as the primary writer.
- **Monitor synchronization health** – Use CloudWatch metrics to track the status of synchronization between your file system and S3 bucket. A growing `PendingExports` indicates that your workload is generating changes faster than the synchronization rate, which means synchronization will take longer to complete. A non-zero `ExportFailures` CloudWatch metric indicates files that could not be exported and require action. For more information, see Troubleshooting S3 Files.

Access control

- **Follow the principle of least privilege** – Grant only the minimum permissions required for each IAM role and file system policy. For example, if a compute resource only needs to read data from the file system, attach the `AmazonS3FilesClientReadOnlyAccess` managed policy instead

of `AmazonS3FilesClientFullAccess`. Additionally, consider creating your file system scoped to a specific prefix rather than the entire bucket, so that clients can only access data within that prefix.

- **Do not modify the S3 Files IAM role** – Do not modify or delete the IAM role that S3 Files assumes to synchronize with your S3 bucket. Changing or removing this role can break synchronization between your file system and S3 bucket.
- **Do not modify the S3 Files EventBridge rule** – S3 Files creates an EventBridge rule (prefixed with DO-NOT-DELETE-S3-Files) to detect changes in your S3 bucket. Do not disable, modify, or delete this rule. Removing it prevents S3 Files from detecting new or changed objects in your bucket, causing your file system to become stale.
- **Consider restricting access to logs written by `efs-utils`** – `efs-utils` writes S3 object key names directly in logs which it stores in the directory `/var/log/amazon/efs`. If your S3 key names contain sensitive information, you should restrict access to this directory through POSIX permissions. For example, you could restrict access through the command `sudo chmod 700 /var/log/amazon/efs`.

Monitoring

- **Set alarms on synchronization failures** – Create CloudWatch alarms on `ImportFailures` and `ExportFailures` to be notified when files fail to synchronize. Failed exports may indicate permission issues, encryption key problems, or path length limits. For more information, see Troubleshooting S3 Files.

Migration

To migrate data from on-premises storage to your S3 bucket for the first time, we recommend using AWS DataSync. DataSync automates and accelerates the transfer of large datasets and preserves file metadata and permissions during migration. For more information, see What is AWS DataSync?.

Unsupported features, limits, and quotas

This page describes the limitations and quotas when using S3 Files.

Unsupported S3 features

- **Archival storage classes** – Objects in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes cannot be accessed through the file system. Similarly, objects in the S3 Intelligent-Tiering Archive Access or Deep Archive Access tiers cannot be accessed through the file system. You must first restore these objects using the S3 API.
- **S3 Access Control Lists (ACLs)** – S3 ACLs are not preserved after changes are made through the file system. We recommend that you keep ACLs disabled and use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Managing access with ACLs.

File system limitations

- **Hard links** – S3 Files does not support hard links. Each file in the file system corresponds to a single S3 object key.
- **Path component size** – S3 Files does not support objects where any single directory or file name in the path exceeds 255 bytes. A path component is a directory or file name derived from an S3 object key. For example, the key `dir1/dir2/file.txt` has three path components: `dir1`, `dir2`, and `file.txt`.
- **Incompatible S3 object key names** – S3 Files does not support accessing S3 key names that do not map to valid POSIX file paths, including empty path components (`foo//bar`), relative path components (`foo/./bar`, `foo/../bar`), and key names containing null bytes.
- **S3 key size limit** – Files or directories whose full path name exceeds the 1,024-byte S3 object key size limit cannot be exported to S3.
- **Symlink targets** – If an S3 object is a symlink, its target must be a valid path that is not empty and does not exceed 4,080 bytes.
- **POSIX permissions metadata size** – Files or directories where the POSIX permission metadata exceeds 2 KB cannot be exported to S3.

Quotas

The following quotas can be increased by contacting AWS Support.

Resource	Default quota
File systems per AWS account	1,000

To request an increase, open the AWS Support Center, choose **Create Case**, then choose **Service Limit Increase**.

The following quotas cannot be changed.

Resource	Quota
Access points per file system	25,000
Connections per file system	25,000
Mount targets per file system per Availability Zone	1
Security groups per mount target	5
Tags per file system	50
VPCs per file system	1

There are also quotas specific to individual file systems.

Resource	Quota
Maximum file size	52,673,613,135,872 bytes (48 TiB)
Maximum directory depth	1,000 levels
Maximum file name length	255 bytes
Maximum symlink target length	4,080 bytes
Maximum S3 object key length	1,024 bytes

Resource	Quota
Maximum open files per client	65,536
Maximum active user accounts per client	65,536
Maximum locks per file	512 across all connected instances
Maximum locks per mount	8,192 across up to 256 file-process pairs
File system policy size limit	20,000 characters

Unsupported NFS features

S3 Files supports NFSv4.1 and NFSv4.2, with the following exceptions:

- pNFS
- Client delegation or callbacks
- Mandatory locking (all locks are advisory)
- Deny share
- Access control lists (ACLs)
- Kerberos-based security
- NFSv4.1 data retention
- SetUID on directories
- Block devices, character devices, attribute directories, and named attributes
- Optional features introduced in NFSv4.2 (such as server-side copy, sparse file operations, space reservation, application I/O advise, application data blocks, labeled NFS, and custom user-defined extended attributes)
- Persistent reply cache
- The `nconnect` mount option

Troubleshooting S3 Files

This page helps you diagnose and resolve common issues with S3 Files.

- Mount command fails
- Permission denied on file operations
- Intelligent read routing is not working
- File system consistently returns NFS server error
- Missing object in S3 bucket after file system write
- S3 object not visible in the file system
- Files appearing in the lost and found directory
- Synchronization falling behind
- File system status is MISCONFIGURED
- Enabling client debug logs
- Error messages

Mount command fails

The `mount -t s3files` command fails with an error.

Common causes and actions:

- **"mount.s3files: command not found"** – The S3 Files client (amazon-efs-utils) is not installed or is below version 3.0.0. Install or upgrade the client. For more information, see Prerequisites for S3 Files.
- **"Failed to resolve file system DNS name"** – There is no mount target in the Availability Zone where your EC2 instance is running. Create a mount target in that Availability Zone, or launch your instance in an Availability Zone that has a mount target. For more information, see Creating mount targets.
- **Connection timed out** – The security group configuration is not allowing NFS traffic. Verify that the mount target's security group allows inbound TCP on port 2049 from your instance's security group, and that your instance's security group allows outbound TCP on port 2049 to the mount target's security group. For more information, see Prerequisites for S3 Files.
- **"Access denied" during mount** – The IAM role attached to your compute resource does not have the required S3 Files permissions. Verify that the role has the

`AmazonS3FilesClientFullAccess` or `AmazonS3FilesClientReadOnlyAccess` managed policy attached, or at minimum the `s3files:ClientMount` permission. For more information, see Prerequisites for S3 Files.

- **botocore not installed** – The mount helper requires botocore to interact with AWS services. Install botocore following the instructions in the amazon-efs-utils README on GitHub.

Permission denied on file operations

You can mount the file system but receive "Permission denied" or "Operation not permitted" errors when reading, writing, or accessing files.

Common causes and actions:

- **Missing write permission** – If you can read but not write, verify that the IAM role attached to your compute resource includes the `s3files:ClientWrite` permission, or attach the `AmazonS3FilesClientReadWriteAccess` or `AmazonS3FilesClientFullAccess` managed policy. For more information, see AWS managed policies for Amazon S3 Files.
- **Missing root access** – If you receive permission errors when accessing files owned by root (UID 0), your IAM role may not have the `s3files:ClientRootAccess` permission. Without this permission, all operations are performed as the NFS anonymous user (typically nfsnobody), which may not have access to the files. Attach the `AmazonS3FilesClientFullAccess` managed policy or add `s3files:ClientRootAccess` to your policy.
- **File system policy denying access** – If you have attached a file system policy, verify that it does not deny the actions your clients need. An "allow" in either the identity-based policy or the file system policy is sufficient for access. For more information, see How S3 Files works with IAM.
- **POSIX permission mismatch** – S3 Files enforces standard POSIX permissions (owner, group, others) on files and directories. If your application runs as a user that does not match the file's owner or group, access may be denied even if IAM permissions are correct. Use an access point to enforce a specific UID/GID for all requests. For more information, see Creating access points for an S3 file system.

Intelligent read routing is not working

S3 Files performs intelligent read routing as it automatically routes read requests to the storage layer best suited for them, while maintaining full file system semantics including consistency, locking, and POSIX permissions. Small, random reads of actively used files are served from the

high-performance storage for low latency. Large sequential reads and reads of data not on the file system are served directly from your S3 bucket for high throughput, with no file system data charge.

Intelligent read routing may not be working if one of the client connectivity metrics (`NFSConnectionAccessible`, `S3BucketAccessible`, and `S3BucketReachable`) shows 0, or if you are not seeing the expected read throughput.

Common causes and actions:

- **Missing S3 inline policy on compute role** – The IAM role attached to your compute resource must include an inline policy granting `s3:GetObject` and `s3:GetObjectVersion` on the linked S3 bucket. Without this policy, the mount helper cannot read directly from S3 and all reads go through the file system. For more information, see Prerequisites for S3 Files.
- **S3 bucket not reachable** – Check the `S3BucketReachable` metric. If it shows 0, verify that your compute resource has network access to S3 (for example, through a VPC endpoint or NAT gateway).
- **File has been modified** – Reads are only served directly from S3 when the file has not been modified through the file system. If you have written to the file and the changes have not yet been synchronized to S3, reads go through the file system until synchronization completes.

File system consistently returns NFS server error

An encrypted file system consistently returns NFS server errors. These errors can occur when S3 Files cannot retrieve your KMS key from AWS KMS for one of the following reasons:

- The key was disabled.
- The key was deleted.
- Permission for S3 Files to use the key was revoked.
- AWS KMS is temporarily unavailable.

Action to take

First, confirm that the AWS KMS key is enabled. You can view your keys in the AWS KMS console. For more information, see Viewing Keys in the *AWS Key Management Service Developer Guide*.

If the key is not enabled, enable it. For more information, see Enabling and Disabling Keys in the *AWS Key Management Service Developer Guide*.

If the key is pending deletion, cancel the deletion and re-enable the key. For more information, see Scheduling and Canceling Key Deletion in the *AWS Key Management Service Developer Guide*.

If the key is enabled and you are still experiencing issues, contact AWS Support.

Missing object in S3 bucket after file system write

You wrote a file through the file system and expected it to appear as an object in your S3 bucket, but the object is not there. S3 Files waits for a period of write inactivity (60 seconds) before exporting changes back to your S3 bucket. If the object still does not appear after this period, the export might have failed. In such a case, you see the `FailedExports` CloudWatch metric increase.

Action to take

Check the file's export status using extended attributes:

```
getfattr -n "user.s3files.status;$(date -u +%s)" missing-file.txt --only-values
```

The timestamp in the attribute name ensures you get the latest status. Example output:

```
S3Key: s3://bucket/prefix/missing-file.txt
ExportError: PathTooLong
```

`ExportError` is not displayed if there is no export failure. `S3Key` is empty if an S3 object was never linked to the file.

The following table lists all possible `ExportError` values:

Error	Cause
`S3AccessDenied`	The IAM role that S3 Files assumes does not have sufficient permissions to write to the S3 bucket. For more information, see Prerequisites for S3 Files.
`InternalError`	There was an internal system error.

Error	Cause
`S3UserMetadataTooLarge`	S3 user metadata size limit exceeded. See Unsupported features, limits, and quotas for information on these limits.
`EncryptionKeyInaccessible`	The encryption key used by the S3 bucket is inaccessible to S3 Files. Grant S3 Files access to your encryption key. For more information, see Encryption.
`RoleAssumptionFailed`	Could not assume the role. Check your trust policies. For more information, see Prerequisites for S3 Files.
`KeyTooLongToBreakCycle`	S3 Files could not resolve a circular dependency (for example, due to renaming two files to each other's names) because the file path exceeds the S3 key length limit. Shorten the directory path to resolve this error.
`PathTooLong`	Your file path exceeds the S3 key length limit. See Unsupported features, limits, and quotas for information on these limits.
`DependencyExportFailed`	A parent or a dependency has a non-retryable export failure. Check the status for the parent or any dependencies using `getfattr`.
`S3ObjectArchived`	S3 object is archived (S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive) and cannot be read. Restore the object first using the S3 APIs.

S3 Files automatically retries failed exports. `ExportError` is shown only for non-retryable errors.

S3 object not visible in the file system

An object exists in your S3 bucket but does not appear in the file system. The object key name may not map to a valid POSIX file path. S3 Files does not support accessing S3 key names with empty path components (`foo//bar`), relative path components (`foo/./bar`, `foo/../bar`), key names containing null bytes, or key names where any path component exceeds 255 bytes. Objects with incompatible key names are not imported into the file system.

Files appearing in the lost and found directory

Files have appeared in the `.s3files-lost+found-`*`file-system-id`* directory in your file system's root directory. In this case, you see the `LostAndFoundFiles` CloudWatch metric increase. This occurs when a synchronization conflict arises. A conflict occurs when the same file is modified through the file system and the corresponding S3 object changes before S3 Files synchronizes the file system changes back to S3. S3 Files treats the S3 bucket as the source of truth, moves the conflicting file to the lost and found directory, and imports the latest version from the S3 bucket into the file system.

Identifying files in the lost and found directory

When S3 Files moves a file to the lost and found directory, it prepends the file name with a hexadecimal identifier to distinguish multiple versions of the same file that may be moved over time. File names longer than 100 characters are truncated to make room for this identifier. The file's original directory path is not preserved in the lost and found directory.

Action to take

Get the file's original path and the corresponding S3 object key:

```
getfattr -n "user.s3files.status;$(date -u +%s)" .s3files-lost
+found-fs-12345678/abcdef1234_report.csv --only-values
```

Example output:

```
S3Key: s3://bucket/prefix/report.csv
FilePath: /data/report.csv
```

Field	Description
`S3Key`	Full S3 path of the object that caused the conflict, or empty if the object was deleted in the S3 bucket.
`FilePath`	Relative path of the file before the conflict.

You can then either keep the latest version from your S3 bucket and delete the file from the lost and found directory, or copy the file from the lost and found directory back to its original path to overwrite the S3 version.

> **Note**
>
> Files in the lost and found directory remain there indefinitely and count toward your file system storage costs. Delete files from the lost and found directory when they are no longer needed.

Synchronization falling behind

The `PendingExports` CloudWatch metric is growing, indicating that your workload is generating changes faster than S3 Files can synchronize them to S3.

This means that your workload may be exceeding the synchronization rate. S3 Files exports up to 800 files per second per file system. Consider reducing the rate of file modifications or distributing work across multiple file systems. Monitor the `PendingExports` metric over time. If it stabilizes or decreases, S3 Files is catching up. If it continues to grow, contact AWS Support.

File system status is MISCONFIGURED

If your file system status is MISCONFIGURED, insufficient permissions prevent S3 Files from importing or exporting data between your file system and your S3 bucket. Changes don't synchronize until you correct the configuration.

The common causes are:

- S3 Files can't assume the AWS Identity and Access Management (IAM) role. Make sure that the trust policy of the IAM role allows S3 Files to assume the role. If the role was deleted or renamed, recreate the role or update your file system to use the correct role.
- The IAM role doesn't have permission to access the S3 bucket. Make sure that the IAM role has a policy that grants access to the linked S3 bucket.
- A bucket policy or a service control policy (SCP) denies access. Make sure that your S3 bucket policy and any AWS Organizations SCPs don't deny access to the IAM role of your file system.
- S3 Files doesn't have permission for a specific key or prefix. If some of your data imports and exports successfully, S3 Files can be missing permission for only part of the bucket under the prefix of your file system. Check the IAM role of your file system, the S3 bucket policy, and the Object Ownership settings of the bucket.
- S3 Files can't use the KMS keys that encrypt your objects. If your bucket uses AWS Key Management Service (AWS KMS) encryption, or if some objects use a different encryption key

than the bucket default, make sure that the key policy of each key allows the IAM role of your file system to use the key. For more information, see S3 Files key policies for AWS KMS.

For more information, see IAM role for accessing your bucket from the file system and How S3 Files works with IAM.

After you correct the configuration, S3 Files automatically retries failed imports and exports. When all failed imports and exports complete, the status of your file system changes to AVAILABLE. It can take more than 24 hours for the status to change to AVAILABLE after you correct the configuration.

> **Important**
>
> If your file system has the MISCONFIGURED status for more than 60 days, some data can become permanently out of sync with your S3 bucket. Correct the configuration as soon as possible. If this occurs, contact AWS Support.

Enabling client debug logs

If you are troubleshooting mount, connectivity, or read bypass issues, you can enable debug-level logging on the S3 Files client to capture more detail.

Mount helper and watchdog logs

Edit `/etc/amazon/efs/s3files-utils.conf` and change the logging level from INFO to DEBUG:

```
[DEFAULT]
logging_level = DEBUG
```

Unmount and remount the file system for the change to take effect:

```
sudo umount /mnt/s3files
sudo mount -t s3files file-system-id:/ /mnt/s3files
```

Logs are written to `/var/log/amazon/efs/`. The mount helper log is `mount.log`.

Proxy (efs-proxy) logs

The proxy handles NFS traffic and S3 read bypass. To enable debug logging for the proxy, edit `/etc/amazon/efs/s3files-utils.conf`:

```
[proxy]
proxy_logging_level = DEBUG
```

Unmount and remount for the change to take effect. Proxy logs are written to `/var/log/amazon/efs/`.

TLS tunnel (stunnel) logs

TLS tunnel logs are disabled by default. To enable them, edit `/etc/amazon/efs/s3files-utils.conf` and set the following:

```
[mount]
stunnel_debug_enabled = true
```

To save all stunnel logs for a file system to a single file, also uncomment the `stunnel_logs_file` line:

```
stunnel_logs_file = /var/log/amazon/efs/{fs_id}.stunnel.log
```

Log size limits

Log files are rotated automatically. You can configure the maximum size and number of rotated files in `s3files-utils.conf`:

```
[DEFAULT]
logging_max_bytes = 1048576
logging_file_count = 10
```

The default is 1 MB per log file with 10 rotated files, for a maximum of 10 MB per log type.

Sharing logs with AWS Support

When contacting AWS Support, collect the client logs and configuration into a single archive:

```
sudo tar -czf /tmp/s3files-support-logs.tar.gz \
  /var/log/amazon/efs/ \
```

```
/etc/amazon/efs/s3files-utils.conf
```

Include `/tmp/s3files-support-logs.tar.gz` with your support case.

Error messages

The following table lists error messages you might encounter and the recommended action for each.

Message	Action Required
Access denied: The provided role does not have permission to call `s3:HeadObject` on the provided bucket.	Your file system IAM role doesn't have sufficient permissions to sync to your bucket. Update your IAM policy to include all required S3 Files permissions. For more information, see Prerequisites for S3 Files.
You attempted to create more file systems than are allowed for an account. For more information, see Unsupported features, limits, and quotas.	For information about how to request a limit increase, see Unsupported features, limits, and quotas.

Access control in Amazon S3

In AWS, a resource is an entity that you can work with. In Amazon Simple Storage Service (S3), *buckets* and *objects* are the original Amazon S3 resources. Every S3 customer likely has buckets with objects in them. As new features were added to S3, additional resources were also added, but not every customer uses these feature-specific resources. For more information about Amazon S3 resources, see S3 resources.

By default, all Amazon S3 resources are private. Also by default, the root user of the AWS account that created the resource (resource owner) and IAM users within that account with the necessary permissions can access a resource that they created. The resource owner decides who else can access the resource and the actions that others are allowed to perform on the resource. S3 has various access management tools that you can use to grant others access to your S3 resources.

The following sections provide you with an overview of S3 resources, the S3 access management tools available, and the best use cases for each access management tool. The lists in these sections aim to be comprehensive and include all S3 resources, access management tools, and common access management use cases. At the same time, these sections are designed to be directories that lead you to the technical details you want. If you have a good understanding of some of the following topics, you can skip to the section that applies to you.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Topics

- S3 resources
- Identities
- Access management tools
- Actions
- Access management use cases
- Access management troubleshooting

S3 resources

The original Amazon S3 resources are buckets and the objects that they contain. As new features are added to S3, new resources are also added. The following is a complete list of S3 resources and their respective features.

Resource type	Amazon S3 feature	Description
`bucket`	Core features	A bucket is a container for objects. To store an object in S3, create a bucket and then upload one or more objects to the bucket. For more information, see Creating, configuring, and working with Amazon S3 general purpose buckets.
`object`		An object can be a file and any metadata that describes that file. When an object is in the bucket, you can open it, download it, and move it. For more information, see Working with objects in Amazon S3.
`accesspoint`	Access Points	Access Points are named network endpoints that are attached to buckets that you can use to perform Amazon S3 **object** operations, such as `GetObject` and `PutObject`. Each access point has distinct permissions, network controls, and a customized *access point policy* that works in conjunction with the bucket policy that is attached to the underlying bucket. You can configure any access point to accept requests only from a virtual private cloud (VPC) or configure custom block public access settings for each access point. For more information, see Managing access to shared datasets with access points.
`objectlambdaaccesspoint`		An Object Lambda Access Point is an access point for a bucket that is also associated with a Lambda function. With Object Lambda Access Point, you can add your own code to Amazon S3 `GET`, `LIST`, and `HEAD` requests to modify and process data as it's returned to an application.

Resource type	Amazon S3 feature	Description
		For more information, see Creating Object Lambda Access Points.
`multiregionaccesspoint`		Multi-Region Access Points provide a global endpoint that applications can use to fulfill requests from Amazon S3 buckets that are located in multiple AWS Regions. You can use Multi-Region Access Points to build multi-Region applications with the same architecture that's used in a single Region, and then run those applications anywhere in the world. Instead of sending requests over the congested public internet, application requests made to a Multi-Region Access Point global endpoint automatically route through the AWS global network to the closest proximity Amazon S3 bucket. For more information, see Managing multi-Region traffic with Multi-Region Access Points.
`job`	S3 Batch Operations	A job is a resource of the S3 Batch Operations feature. You can use S3 Batch Operations to perform large-scale batch operations on lists of Amazon S3 objects that you specify. Amazon S3 tracks the progress of the batch operation job, sends notifications, and stores a detailed completion report of all actions, providing you with a fully managed, auditable, and serverless experience. For more information, see Performing object operations in bulk with Batch Operations.

Resource type	Amazon S3 feature	Description
`storagelensconfiguration`	S3 Storage Lens	An S3 Storage Lens configuration collects organization-wide storage metrics and user data across accounts. S3 Storage Lens provides admins with a single view of object storage usage and activity across hundreds, or even thousands, of accounts in an organization, with details to generate insights at multiple aggregation levels. For more information, see Monitoring your storage activity and usage with Amazon S3 Storage Lens.
`storagelensgroup`		An S3 Storage Lens group aggregates metrics by using custom filters based on object metadata. S3 Storage Lens groups help you investigate characteristics of your data, such as distribution of objects by age, your most common file types, and more. For more information, see Working with S3 Storage Lens groups to filter and aggregate metrics.
`accessgrantsinstance`	S3 Access Grants	An S3 Access Grants instance is a container for the S3 grants that you create. With S3 Access Grants, you can create grants to your Amazon S3 data for IAM identities within your account, IAM identities in other accounts (cross-account), and directory identities added to AWS IAM Identity Center from your corporate directory. For more information about S3 Access Grants, see Managing access with S3 Access Grants.

Resource type	Amazon S3 feature	Description
`accessgrantslocation`		An Access Grants Location is a bucket, prefix within a bucket, or an object that you register in your S3 Access Grants instance. You must register locations within the S3 Access Grants instance before you can create a grant to that location. Then, with S3 Access Grants, you can grant access to the bucket, prefix, or object for IAM identities within your account, IAM identities in other accounts (cross-account), and directory identities added to AWS IAM Identity Center from your corporate directory. For more information about S3 Access Grants, see Managing access with S3 Access Grants
`accessgrant`		An Access Grant is an individual grant to your Amazon S3 data. With S3 Access Grants, you can create grants to your Amazon S3 data for IAM identities within your account, IAM identities in other accounts (cross-account), and directory identities added to AWS IAM Identity Center from your corporate directory. For more information about S3 Access Grants, see Managing access with S3 Access Grants

Buckets

There are two types of Amazon S3 buckets: *general purpose buckets* and *directory buckets*.

- **General purpose buckets** are the original S3 bucket type and are recommended for most use cases and access patterns. General purpose buckets also allow objects that are stored across all storage classes, except S3 Express One Zone. For more information about S3 storage classes, see Understanding and managing Amazon S3 storage classes.
- **Directory buckets** use the S3 Express One Zone storage class, which is recommended if your application is performance-sensitive and benefits from single-digit millisecond PUT and GET latencies. For more information, see Working with directory buckets, S3 Express One Zone, and Authorizing Regional endpoint API operations with IAM.

Categorizing S3 resources

Amazon S3 provides features to categorize and organize your S3 resources. Categorizing your resources is not only useful for organizing them, but you can also set access management rules based on the resource categories. In particular, prefixes and tagging are two storage organization features that you can use when setting access management permissions.

> **Note**
>
> The following information applies to general purpose buckets. Directory buckets do not support tagging, and they have prefix limitations. For more information, see Authorizing Regional endpoint API operations with IAM.

- **Prefixes** — A prefix in Amazon S3 is a string of characters at the beginning of an object key name that's used to organize the objects that are stored in your S3 buckets. You can use a delimiter character, such as a forward slash (/), to indicate the end of the prefix within the object key name. For example, you might have object key names that start with the `engineering/` prefix or object key names that start with the `marketing/campaigns/` prefix. Using a delimeter at the end of your prefix, such as as a forward slash character `/` emulates folder and file naming conventions. However, in S3, the prefix is part of the object key name. In general purpose S3 buckets, there is no actual folder hierarchy.

  Amazon S3 supports organizing and grouping objects by using their prefixes. You can also manage access to objects by their prefixes. For example, you can limit access to only the objects with names that start with a specific prefix.

  For more information, see Organizing objects using prefixes. S3 Console uses the concept of *folders*, which, in general purpose buckets, are essentially prefixes that are pre-pended to the object key name. For more information, see Organizing objects in the Amazon S3 console by using folders.
- **Tags** — Each tag is a key-value pair that you assign to resources. For example, you can tag some resources with the tag `topicCategory=engineering`. You can use tagging to help with cost allocation, categorizing and organizing, and access control. Bucket tagging is only used for cost allocation. You can tag objects, S3 Storage Lens, jobs, and S3 Access Grants for the purposes of organizing or for access control. In S3 Access Grants, you can also use tagging for cost-allocation. As an example of controlling access to resources by using their tags, you can share only the objects that have a specific tag or a combination of tags.

For more information, see Controlling access to AWS resources by using resource tags in the *IAM User Guide*.

Identities

In Amazon S3, the resource owner is the identity that created the resource, such as a bucket or an object. By default, only the root user of the account that created the resource and IAM identities within the account that have the required permission can access the S3 resource. Resource owners can give other identities access to their S3 resources.

Identities that don't own a resource can request access to that resource. Requests to a resource are either authenticated or unauthenticated. Authenticated requests must include a signature value that authenticates the request sender, but unauthenticated requests do not require a signature. We recommend that you grant access only to authenticated users. For more information about request authentication, see Making requests in the *Amazon S3 API Reference*.

Important

We recommend that you don't use the AWS account root user credentials to make authenticated requests. Instead, create an IAM role and grant that role full access. We refer to users with this role as *administrator users*. You can use credentials assigned to the administrator role, instead of AWS account root user credentials, to interact with AWS and perform tasks, such as create a bucket, create users, and grant permissions. For more information, see AWS account root user credentials and IAM user credentials in the *AWS General Reference*, and see Security best practices in IAM in the *IAM User Guide*.

Identities accessing your data in Amazon S3 can be one of the following:

AWS account owner

The AWS account that created the resource. For example, the account that created the bucket. This account owns the resource. For more information, see AWS account root user.

IAM identities in the same account of the AWS account owner

When setting up accounts for new team members who require S3 access, the AWS account owner can use AWS Identity and Access Management (IAM) to create users, groups, and roles. The AWS

account owner can then share resources with these IAM identities. The account owner can also specify the permissions to give the IAM identities, which allow or deny the actions that can be performed on the shared resources.

IAM identities provide increased capabilities, including the ability to require users to enter login credentials before accessing shared resources. By using IAM identities, you can implement a form of IAM multi-factor authentication (MFA) to support a strong identity foundation. An IAM best practice is to create roles for access management instead of granting permissions to each individual user. You assign individual users to the appropriate role. For more information, see Security best practices in IAM.

Other AWS account owners and their IAM identities (cross-account access)

The AWS account owner can also give other AWS account owners, or IAM identities that belong to another AWS account, access to resources.

> **Note**
>
> **Permission delegation** — If an AWS account owns a resource, it can grant those permissions to another AWS account. That account can then delegate those permissions, or a subset of them, to users in the same account. This is referred to as permission delegation. But an account that receives permissions from another account cannot delegate those permissions "cross-account" to another AWS account.

Anonymous users (public access)

The AWS account owner can make resources public. Making a resource public technically shares the resource with *the anonymous user*. Buckets created since April 2023 block all public access by default, unless you change this setting. We recommend that you set your buckets to block public access, and that you only grant access to authenticated users. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

AWS services

The resource owner can grant another AWS service access to an Amazon S3 resource. For example, you can grant the AWS CloudTrail service `s3:PutObject` permission to write log files to your bucket. For more information, see Providing access to an AWS service.

Corporate directory identities

The resource owner can grant users or roles from your corporate directory access to an S3 resource by using S3 Access Grants. For more information about adding your corporate directory to AWS IAM Identity Center, see What is IAM Identity Center?.

Bucket or resource owners

The AWS account that you use to create buckets and upload objects owns those resources. A bucket owner can grant cross-account permissions to another AWS account (or users in another account) to upload objects.

When a bucket owner permits another account to upload objects to a bucket, the bucket owner, by default, owns all objects uploaded to their bucket. However, if both the *Bucket owner enforced* and *Bucket owner preferred* bucket settings are turned off, the AWS account that uploads the objects owns those objects, and the bucket owner does not have permissions on the objects owned by another account, with the following exceptions:

- The bucket owner pays the bills. The bucket owner can deny access to any objects, or delete any objects in the bucket, regardless of who owns them.
- The bucket owner can archive any objects or restore archived objects, regardless of who owns them. Archival refers to the storage class used to store the objects. For more information, see Managing the lifecycle of objects.

Access management tools

Amazon S3 provides a variety of security features and tools. The following is a comprehensive list of these features and tools. You do not need all of these access management tools, but you must use one or more to grant access to your Amazon S3 resources. Proper application of these tools can help make sure that your resources are accessible only to the intended users.

The most commonly used access management tool is an *access policy*. An access policy can be a *resource-based policy* that is attached to an AWS resource, such as a bucket policy for a bucket. An access policy can also be an *identity-based policy* that is attached to an AWS Identity and Access Management (IAM) identity, such as an IAM user, group, or role. Write an access policy to grant AWS accounts and IAM users, groups, and roles permission to perform operations on a resource. For example, you can grant `PUT Object` permission to another AWS account so that the other account can upload objects to your bucket.

An access policy describes who has access to what things. When Amazon S3 receives a request, it must evaluate all of the access policies to determine whether to authorize or deny the request. For more information about how Amazon S3 evaluates these policies, see How Amazon S3 authorizes a request.

The following are the access management tools available in Amazon S3.

Bucket policy

An Amazon S3 bucket policy is a JSON-formatted AWS Identity and Access Management (IAM) resource-based policy that is attached to a particular bucket. Use bucket policies to grant other AWS accounts or IAM identities permissions for the bucket and the objects in it. Many S3 access management use cases can be met by using a bucket policy. With bucket policies, you can personalize bucket access to help make sure that only the identities that you have approved can access resources and perform actions within them. For more information, see Bucket policies for Amazon S3.

The following is an example bucket policy. You express the bucket policy by using a JSON file. This example policy grants an IAM role read permission to all objects in the bucket. It contains one statement named `BucketLevelReadPermissions`, which allows the `s3:GetObject` action (read permission) on objects in a bucket named `amzn-s3-demo-bucket1`. By specifying an IAM role as the `Principal`, this policy grants access to any IAM user with this role. To use this example policy, replace the *`user input placeholders`* with your own information.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":"BucketLevelReadPermissions",
      "Effect":"Allow",
      "Principal": {
     "AWS": "arn:aws:iam::123456789101:role/s3-role"
      },
      "Action":["s3:GetObject"],
      "Resource":["arn:aws:s3:::amzn-s3-demo-bucket/*"]
    }]
  }
```

Note

When creating policies, avoid the use of wildcard characters (*) in the `Principal` element because using a wildcard character allows anyone to access your Amazon S3 resources. Instead, explicitly list users or groups that are allowed to access the bucket, or list conditions that must be met by using a condition clause in the policy. Also, rather than including a wildcard character for the actions of your users or groups, grant them specific permissions when applicable.

Identity-based policy

An identity-based or IAM user policy is a type of AWS Identity and Access Management (IAM) policy. An identity-based policy is a JSON-formatted policy that is attached to IAM users, groups, or roles in your AWS account. You can use identity-based policies to grant an IAM identity access to your buckets or objects. You can create IAM users, groups, and roles in your account and attach access policies to them. You can then grant access to AWS resources, including Amazon S3 resources. For more information, see Identity-based policies for Amazon S3.

The following is an example of an identity-based policy. The example policy allows the associated IAM role to perform six different Amazon S3 actions (permissions) on a bucket and the objects in it. If you attach this policy to an IAM role in your account and assign the role to some of your IAM users, the users with this role will be able to perform these actions on the resources (buckets) specified in your policy. To use this example policy, replace the *`user input placeholders`* with your own information.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
  {
    "Sid": "AssignARoleActions",
    "Effect": "Allow",
    "Action": [
   "s3:PutObject",
   "s3:GetObject",
   "s3:ListBucket",
   "s3:DeleteObject",
```

```
    "s3:GetBucketLocation"
     ],
     "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "arn:aws:s3:::amzn-s3-demo-bucket"
     ]
     },
   {
     "Sid": "AssignARoleActions2",
     "Effect": "Allow",
     "Action": "s3:ListAllMyBuckets",
     "Resource": "*"
   }
   ]
}
```

S3 Access Grants

Use S3 Access Grants to create access grants to your Amazon S3 data for both identities in corporate identity directories, such as Active Directory, and to AWS Identity and Access Management (IAM) identities. S3 Access Grants helps you manage data permissions at scale. Additionally, S3 Access Grants logs end-user identity and the application used to access the S3 data in AWS CloudTrail. This provides a detailed audit history down to the end-user identity for all access to the data in your S3 buckets. For more information, see Managing access with S3 Access Grants.

Access Points

Amazon S3 Access Points simplifies managing data access at scale for applications that use shared datasets on S3. Access Points are named network endpoints that are attached to a bucket. You can use access points to perform S3 object operations at scale, such as uploading and retrieving objects. A bucket can have up to 10,000 access points attached, and for each access point, you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. S3 Access Points can be associated with buckets in the same account or in another trusted account. Access Points policies are resource-based policies that are evaluated in conjunction with the underlying bucket policy. For more information, see Managing access to shared datasets with access points.

Access control list (ACL)

An ACL is a list of grants identifying the grantee and the permission granted. ACLs grant basic read or write permissions to other AWS accounts. ACLs use an Amazon S3–specific XML schema. An ACL is a type of AWS Identity and Access Management (IAM) policy. An object ACL is used to manage access to an object, and a bucket ACL is used to manage access to a bucket. With bucket policies, there is a single policy for the entire bucket, but object ACLs are specified for each object. We recommend that you keep ACLs turned off, except in circumstances where you must individually control access for each object. For more information about using ACLs, see Controlling ownership of objects and disabling ACLs for your bucket.

> **Warning**
>
> The majority of modern use cases in Amazon S3 do not require the use of ACLs.

The following is an example bucket ACL. The grant in the ACL shows a bucket owner that has full control permission.

```
<?xml version="1.0" encoding="UTF-8"?>
 <AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
   <ID>Owner-Canonical-User-ID</ID>
  </Owner>
 <AccessControlList>
  <Grant>
   <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Canonical
 User">
    <ID>Owner-Canonical-User-ID</ID>
   </Grantee>
   <Permission>FULL_CONTROL</Permission>
  </Grant>
 </AccessControlList>
</AccessControlPolicy>
```

Object Ownership

To manage access to your objects, you must be the owner of the object. You can use the Object Ownership bucket-level setting to control ownership of objects uploaded to your bucket. Also, use Object Ownership to turn on ACLs. By default, Object Ownership is set to the *Bucket owner*

enforced setting and all ACLs are turned off. When ACLs are turned off, the bucket owner owns all of the objects in the bucket and exclusively manages access to data. To manage access, the bucket owner uses policies or another access management tool, excluding ACLs. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

Object Ownership has three settings that you can use both to control ownership of objects that are uploaded to your bucket and to turn on ACLs:

ACLs turned off

- **Bucket owner enforced (default)** – ACLs are turned off, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs do not affect permissions to data in the S3 bucket. The bucket uses policies exclusively to define access control.

ACLs turned on

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

Additional best practices

Consider using the following bucket settings and tools to help protect data in transit and at rest, both of which are crucial in maintaining the integrity and accessibility of your data:

- **Block Public Access** — Do not turn off the default bucket-level setting *Block Public Access*. This setting blocks public access to your data by default. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.
- **S3 Versioning** — For data integrity, you can implement the S3 Versioning bucket setting, which versions your objects as you make updates, instead of overwriting them. You can use S3 Versioning to preserve, retrieve, and restore a previous version, if needed. For information about S3 Versioning, see Retaining multiple versions of objects with S3 Versioning.
- **S3 Object Lock** — S3 Object Lock is another setting that you can implement for achieving data integrity. This feature can implement a write-once-read-many (WORM) model to store objects immutably. For information about Object Lock, see Locking objects with Object Lock.

- **Object encryption** — Amazon S3 offers several object encryption options that protect data in transit and at rest. *Server-side encryption* encrypts your object before saving it on disks in its data centers and then decrypts it when you download the objects. If you authenticate your request and you have access permissions, there is no difference in the way you access encrypted or unencrypted objects. For more information, see Protecting data with server-side encryption. S3 encrypts newly uploaded objects by default. For more information, see Setting default server-side encryption behavior for Amazon S3 buckets. *Client-side encryption* is the act of encrypting data before sending it to Amazon S3. For more information, see Protecting data by using client-side encryption.
- **Signing methods** — Signature Version 4 is the process of adding authentication information to AWS requests sent by HTTP. For security, most requests to AWS must be signed with an access key, which consists of an access key ID and secret access key. These two keys are commonly referred to as your security credentials. For more information, see Authenticating Requests (AWS Signature Version 4) and Signature Version 4 signing process.

Actions

For a complete list of S3 permissions and condition keys, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Actions

The AWS Identity and Access Management (IAM) actions for Amazon S3 are the possible actions that can be performed on an S3 bucket or object. You grant these actions to identities so they can act on your S3 resources. Examples of S3 actions are `s3:GetObject` to read objects in a bucket, and `s3:PutObject` to write objects to a bucket.

Condition keys

In addition to actions, IAM condition keys are limited to granting access to only when a condition is met. Condition keys are optional.

Note

In a resource-based access policy, such as a bucket policy, or in an identity-based policy, you can specify the following:

- An action or an array of actions in the `Action` element of the policy statement.
- In the `Effect` element of the policy statement, you can specify `Allow` to grant the actions listed, or you can specify `Deny` to block the listed actions. To further maintain the practice of least privileges, `Deny` statements in the `Effect` element of the access policy should be as broad as possible, and `Allow` statements should be as narrow as possible. `Deny` effects paired with the `s3:*` action are another good way to implement opt-in best practices for the identities that are included in policy condition statements.
- A condition key in the `Condition` element of a policy statement.

Access management use cases

Amazon S3 provides resource owners with a variety of tools for granting access. The S3 access management tool that you use depends on the S3 resources that you want to share, the identities that you are granting access to, and the actions that you want to allow or deny. You might want to use one or a combination of S3 access management tools to manage access to your S3 resources.

In most cases, you can use an access policy to manage permissions. An access policy can be a resource-based policy, which is attached to a resource, such as a bucket, or another Amazon S3 resource (S3 resources). An access policy can also be an identity-based policy, which is attached to an AWS Identity and Access Management (IAM) user, group, or role in your account. You might find that a bucket policy works better for your use case. For more information, see Bucket policies for Amazon S3. Alternatively, with AWS Identity and Access Management (IAM), you can create IAM users, groups, and roles within your AWS account and manage their access to buckets and objects through identity-based policies. For more information, see Identity-based policies for Amazon S3.

To help you navigate these access management options, the following are common Amazon S3 customer use cases and recommendations for each of the S3 access management tools.

The AWS account owner wants to share buckets only with users within the same account

All access management tools can fulfill this basic use case. We recommend the following access management tools for this use case:

- **Bucket policy** – If you want to grant access to one bucket or a small number of buckets, or if your bucket access permissions are similar from bucket to bucket, use a bucket policy. With bucket policies, you manage one policy for each bucket. For more information, see Bucket policies for Amazon S3.
- **Identity-based policy** – If you have a very large number of buckets with different access permissions for each bucket, and only a few user roles to manage, you can use an IAM policy for users, groups, or roles. IAM policies are also a good option if you are managing user access to other AWS resources, as well as Amazon S3 resources. For more information, see Example 1: Bucket owner granting its users bucket permissions.
- **S3 Access Grants** – You can use S3 Access Grants to grant access to your S3 buckets, prefixes, or objects. S3 Access Grants allows you to specify varying object-level permissions at scale; whereas, bucket policies are limited to 20 KB in size. For more information, see Getting started with S3 Access Grants.
- **Access Points** – You can use Access Points, which are named network endpoints that are attached to a bucket. A bucket can have up to 10,000 access points attached, and for each access point you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. For more information, see Managing access to shared datasets with access points.

The AWS account owner wants to share buckets or objects with users from another AWS account (cross-account)

To grant permission to another AWS account, you must use a bucket policy or one of the following recommended access management tools. You cannot use an identity-based access policy for this use case. For more information about granting cross-account access, see How do I provide cross-account access to objects that are in Amazon S3 buckets?

We recommend the following access management tools for this use case:

- **Bucket policy** – With bucket policies, you manage one policy for each bucket. For more information, see Bucket policies for Amazon S3.

- **S3 Access Grants** – You can use S3 Access Grants to grant cross-account permissions to your S3 buckets, prefixes, or objects. You can use S3 Access Grants to specify varying object-level permissions at scale; whereas, bucket policies are limited to 20 KB in size. For more information, see Getting started with S3 Access Grants.
- **Access Points** – You can use Access Points, which are named network endpoints that are attached to a bucket. A bucket can have up to 10,000 access points attached, and for each access point you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. For more information, see Managing access to shared datasets with access points.

The AWS account owner or bucket owner must grant permissions at the object-level or prefix-level, and these permissions vary from object to object or prefix to prefix

In a bucket policy, for example, you can grant access to the objects within a bucket that share a specific key name prefix or have a specific tag. You can grant read permission on objects starting with the key name prefix `logs/`. However, if your access permissions vary by object, granting permissions to individual objects by using a bucket policy might not be practical, especially since bucket policies are limited to 20 KB in size.

We recommend the following access management tools for this use case:

- **S3 Access Grants** – You can use S3 Access Grants to manage object-level or prefix-level permissions. Unlike bucket policies, you can use S3 Access Grants to specify varying object-level permissions at scale. Bucket policies are limited to 20 KB in size. For more information, see Getting started with S3 Access Grants.
- **Access Points** – You can use access points to manage object-level or prefix-level permissions. Access Points are named network endpoints that are attached to a bucket. A bucket can have up to 10,000 access points attached, and for each access point you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. For more information, see Managing access to shared datasets with access points.
- **ACLs** – We do not recommend using Access Control Lists (ACLs), especially because ACLs are limited to 100 grants per object. However, if you choose to turn on ACLs, in your Bucket Settings, set *Object Ownership* to *Bucket owner preferred* and *ACLs enabled*. With this setting, new objects that are written with the `bucket-owner-full-control` canned ACL are automatically owned by the bucket owner rather than the object writer. You can then use object ACLs, which is an

XML-formatted access policy, to grant other users access to the object. For more information, see Access control list (ACL) overview.

The AWS account owner or bucket owner wants to limit bucket access only to specific account IDs

We recommend the following access management tools for this use case:

- **Bucket policy** – With bucket policies, you manage one policy for each bucket. For more information, see Bucket policies for Amazon S3.
- **Access Points** – Access Points are named network endpoints that are attached to a bucket. A bucket can have up to 10,000 access points attached, and for each access point you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. For more information, see Managing access to shared datasets with access points.

The AWS account owner or bucket owner wants distinct endpoints for every user or application that accesses their data

We recommend the following access management tool for this use case:

- **Access Points** – Access Points are named network endpoints that are attached to a bucket. A bucket can have up to 10,000 access points attached, and for each access point you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. Each access point enforces a customized access point policy that works in conjunction with the bucket policy that is attached to the underlying bucket. For more information, see Managing access to shared datasets with access points.

The AWS account owner or bucket owner must manage access from Virtual Private Cloud (VPC) endpoints for S3

Virtual Private Cloud (VPC) endpoints for Amazon S3 are logical entities within a VPC that allow connectivity only to S3. We recommend the following access management tools for this use case:

- **Buckets in a VPC setting** – You can use a bucket policy to control who is allowed to access your buckets and which VPC endpoints they can access. For more information, see Controlling access from VPC endpoints with bucket policies.

- **Access Points** – If you choose to set up access points, you can use an access point policy. You can configure any access point to accept requests only from a virtual private cloud (VPC) to restrict Amazon S3 data access to a private network. You can also configure custom block public access settings for each access point. For more information, see Managing access to shared datasets with access points.

The AWS account owner or bucket owner must make a static website publicly available

With S3, you can host a static website and allow anyone to view the content of the website, which is hosted from an S3 bucket.

We recommend the following access management tools for this use case:

- **Amazon CloudFront** – This solution allows you to host an Amazon S3 static website to the public while also continuing to block all public access to a bucket's content. If you want to keep all four S3 Block Public Access settings enabled and host an S3 static website, you can use Amazon CloudFront origin access control (OAC). Amazon CloudFront provides the capabilities required to set up a secure static website. Also, Amazon S3 static websites that do not use this solution can only support HTTP endpoints. CloudFront uses the durable storage of Amazon S3 while providing additional security headers, such as HTTPS. HTTPS adds security by encrypting a normal HTTP request and protecting against common cyberattacks.

  For more information, see Getting started with a secure static website in the *Amazon CloudFront Developer Guide*.
- **Making your Amazon S3 bucket publicly accessible** – You can configure a bucket to be used as a publicly accessed static website.

  > **Warning**
  >
  > We do not recommend this method. Instead, we recommend you use Amazon S3 static websites as a part of Amazon CloudFront. For more information, see the previous option, or see Getting started with a secure static website.

  To create an Amazon S3 static website, without Amazon CloudFront, first, you must turn off all Block Public Access settings. When writing the bucket policy for your static website, make sure that you allow only `s3:GetObject` actions, not `ListObject` or `PutObject` permissions. This

helps make sure that users cannot view all the objects in your bucket or add their own content. For more information, see Setting permissions for website access.

The AWS account owner or bucket owner wants to make the content of a bucket publicly available

When creating a new Amazon S3 bucket, the *Block Public Access* setting is enabled by default. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

We do not recommend allowing public access to your bucket. However, if you must do so for a particular use case, we recommend the following access management tool for this use case:

- **Disable Block Public Access setting** – A bucket owner can allow unauthenticated requests to the bucket. For example, unauthenticated PUT Object requests are allowed when a bucket has a public bucket policy, or when a bucket ACL grants public access. All unauthenticated requests are made by other arbitrary AWS users, or even unauthenticated, anonymous users. This user is represented in ACLs by the specific canonical user ID `65a011a29cdf8ec533ec3d1ccaae921c`. If an object is uploaded to a `WRITE` or `FULL_CONTROL`, then this specifically grants access to the All Users group or the anonymous user. For more information about public bucket policies and public access control lists (ACLs), see The meaning of "public".

The AWS account owner or bucket owner has exceeded access policy size limits

Both bucket policies and identity-based policies have a 20 KB size limit. If your access permission requirements are complex, you might exceed this size limit.

We recommended the following access management tools for this use case:

- **Access Points** – Use access points if this works with your use case. With access points, each bucket has multiple named network endpoints, each with its own access point policy that works with the underlying bucket policy. However, access points can only act on objects, not buckets, and does not support cross-Region replication. For more information, see Managing access to shared datasets with access points.
- **S3 Access Grants** – Use S3 Access Grants, which supports a very large number of grants that give access to buckets, prefixes, or objects. For more information, see Getting started with S3 Access Grants.

The AWS account owner or admin role wants to grant bucket, prefix, or object access directly to users or groups in a corporate directory

Instead of managing users, groups, and roles through AWS Identity and Access Management (IAM), you can add your corporate directory to AWS IAM Identity Center. For more information, see What is IAM Identity Center?.

After you add your corporate directory to AWS IAM Identity Center, we recommend that you use the following access management tool to grant corporate directory identities access to your S3 resources:

- **S3 Access Grants** – Use S3 Access Grants, which supports granting access to users or roles in your corporate directory. For more information, see Getting started with S3 Access Grants.

The AWS account owner or bucket owner wants to give the AWS CloudFront service access to write CloudFront logs to an S3 bucket

We recommended the following access management tool for this use case:

- **Bucket ACL** – The only recommended use case for bucket ACLs is to grant permissions to certain AWS services, such as the Amazon CloudFront `awslogsdelivery` account. When you create or update a distribution and turn on CloudFront logging, CloudFront updates the bucket ACL to give the `awslogsdelivery` account `FULL_CONTROL` permissions to write logs to your bucket. For more information, see Permissions required to configure standard logging and to access your log files in the *Amazon CloudFront Developer Guide*. If the bucket that stores the logs uses the *Bucket owner enforced* setting for S3 Object Ownership to turn off ACLs, CloudFront cannot write logs to the bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

You, as the bucket owner, want to maintain full control of objects that are added to the bucket by other users

You can grant other accounts access to upload objects to your bucket by using a bucket policy, access point, or S3 Access Grants. If you have granted cross-account access to your bucket, you can make sure that any objects uploaded to your bucket remain under your full control.

We recommended the following access management tool for this use case:

- **Object Ownership** – Keep the bucket-level setting *Object Ownership* at the default *Bucket owner enforced* setting.

Access management troubleshooting

The following resources can help you troubleshoot any issues with S3 access management:

Troubleshooting Access Denied (403 Forbidden) errors

If you encounter access denial issues, check the account-level and bucket-level settings. Also, check the access management feature that you are using to grant access to make sure that the policy, setting, or configuration is correct. For more information about common causes of Access Denied (403 Forbidden) errors in Amazon S3, see Troubleshoot access denied (403 Forbidden) errors in Amazon S3.

IAM Access Analyzer for S3

If you do not want to make any of your resources publicly available, or if you want to limit public access to your resources, you can use IAM Access Analyzer for S3. On the Amazon S3 console, use IAM Access Analyzer for S3 to review all buckets that have bucket access control lists (ACLs), bucket policies, or access point policies that grant public or shared access. IAM Access Analyzer for S3 alerts you to buckets that are configured to allow access to anyone on the internet or other AWS accounts, including AWS accounts outside of your organization. For each public or shared bucket, you receive findings that report the source and level of public or shared access.

In IAM Access Analyzer for S3, you can block all public access to a bucket with a single action. We recommend that you block all public access to your buckets, unless you require public access to support a specific use case. Before you block all public access, make sure that your applications will continue to work correctly without public access. For more information, see Blocking public access to your Amazon S3 storage.

You can also review your bucket-level permission settings to configure detailed levels of access. For specific and verified use cases that require public or shared access, you can acknowledge and record your intent for the bucket to remain public or shared by archiving the findings for the bucket. You can revisit and modify these bucket configurations at any time. You can also download your findings as a CSV report for auditing purposes.

IAM Access Analyzer for S3 is available at no extra cost on the Amazon S3 console. IAM Access Analyzer for S3 is powered by AWS Identity and Access Management (IAM) IAM Access Analyzer.

To use IAM Access Analyzer for S3 on the Amazon S3 console, you must visit the IAM Console and create an account-level analyzer in IAM Access Analyzer for each individual Region.

For more information about IAM Access Analyzer for S3, see Reviewing bucket access using IAM Access Analyzer for S3.

Logging and monitoring

Monitoring is an important part of maintaining the reliability, availability, and performance of your Amazon S3 solutions so that you can more easily debug an access failure. Logging can provide insight into any errors users are receiving, and when and what requests are made. AWS provides several tools for monitoring your Amazon S3 resources, such as the following:

- AWS CloudTrail
- Amazon S3 Access Logs
- AWS Trusted Advisor
- Amazon CloudWatch

For more information, see Logging and monitoring in Amazon S3.

Identity and Access Management for Amazon S3

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use Amazon S3 resources. IAM is an AWS service that you can use with no additional charge.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Topics

- Audience
- Authenticating with identities
- Managing access using policies
- How Amazon S3 works with IAM
- How Amazon S3 authorizes a request
- Required permissions for Amazon S3 API operations
- Policies and permissions in Amazon S3
- Bucket policies for Amazon S3
- Identity-based policies for Amazon S3
- Walkthroughs that use policies to manage access to your Amazon S3 resources
- Using service-linked roles for Amazon S3 Storage Lens
- Troubleshooting Amazon S3 identity and access
- AWS managed policies for Amazon S3

Audience

How you use AWS Identity and Access Management (IAM) differs based on your role:

- **Service user** - request permissions from your administrator if you cannot access features (see Troubleshooting Amazon S3 identity and access)
- **Service administrator** - determine user access and submit permission requests (see How Amazon S3 works with IAM)
- **IAM administrator** - write policies to manage access (see Identity-based policies for Amazon S3)

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be authenticated as the AWS account root user, an IAM user, or by assuming an IAM role.

You can sign in as a federated identity using credentials from an identity source like AWS IAM Identity Center (IAM Identity Center), single sign-on authentication, or Google/Facebook credentials. For more information about signing in, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

For programmatic access, AWS provides an SDK and CLI to cryptographically sign requests. For more information, see AWS Signature Version 4 for API requests in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity called the AWS account *root user* that has complete access to all AWS services and resources. We strongly recommend that you don't use the root user for everyday tasks. For tasks that require root user credentials, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users to use federation with an identity provider to access AWS services using temporary credentials.

A *federated identity* is a user from your enterprise directory, web identity provider, or Directory Service that accesses AWS services using credentials from an identity source. Federated identities assume roles that provide temporary credentials.

For centralized access management, we recommend AWS IAM Identity Center. For more information, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity with specific permissions for a single person or application. We recommend using temporary credentials instead of IAM users with long-term credentials. For more information, see Require human users to use federation with an identity provider to access AWS using temporary credentials in the *IAM User Guide*.

An *IAM group* specifies a collection of IAM users and makes permissions easier to manage for large sets of users. For more information, see Use cases for IAM users in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity with specific permissions that provides temporary credentials. You can assume a role by switching from a user to an IAM role (console) or by calling an AWS CLI or AWS API operation. For more information, see Methods to assume a role in the *IAM User Guide*.

IAM roles are useful for federated user access, temporary IAM user permissions, cross-account access, cross-service access, and applications running on Amazon EC2. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy defines permissions when associated with an identity or resource. AWS evaluates these policies when a principal makes a request. Most policies are stored in AWS as JSON documents. For more information about JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Using policies, administrators specify who has access to what by defining which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. An IAM administrator creates IAM policies and adds them to roles, which users can then assume. IAM policies define permissions regardless of the method used to perform the operation.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you attach to an identity (user, group, or role). These policies control what actions identities can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

Identity-based policies can be *inline policies* (embedded directly into a single identity) or *managed policies* (standalone policies attached to multiple identities). To learn how to choose between managed and inline policies, see Choose between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples include IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. You must specify a principal in a resource-based policy.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Other policy types

AWS supports additional policy types that can set the maximum permissions granted by more common policy types:

- **Permissions boundaries** – Set the maximum permissions that an identity-based policy can grant to an IAM entity. For more information, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – Specify the maximum permissions for an organization or organizational unit in AWS Organizations. For more information, see Service control policies in the *AWS Organizations User Guide*.
- **Resource control policies (RCPs)** – Set the maximum available permissions for resources in your accounts. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.
- **Session policies** – Advanced policies passed as a parameter when creating a temporary session for a role or federated user. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon S3 works with IAM

Before you use IAM to manage access to Amazon S3, learn what IAM features are available to use with Amazon S3.

IAM features you can use with Amazon S3

IAM feature	Amazon S3 support
Identity-based policies	Yes
Resource-based policies	Yes
Policy actions	Yes
Policy resources	Yes
Policy condition keys (service-specific)	Yes
ACLs	Yes

IAM feature	Amazon S3 support
ABAC (tags in policies)	Partial
Temporary credentials	Yes
Forward access sessions (FAS)	Yes
Service roles	Yes
Service-linked roles	Partial

To get a high-level view of how Amazon S3 and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Identity-based policies for Amazon S3

Supports identity-based policies: Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Define custom IAM permissions with customer managed policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for Amazon S3

To view examples of Amazon S3 identity-based policies, see Identity-based policies for Amazon S3.

Resource-based policies within Amazon S3

Supports resource-based policies: Yes

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that

support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. For more information, see Cross account resource access in IAM in the *IAM User Guide*.

The Amazon S3 service supports *bucket policies*, *access points policies*, and *access grants*:

- Bucket policies are resource-based policies that are attached to an Amazon S3 bucket. A bucket policy defines which principals can perform actions on the bucket.
- Access point policies are resource-based polices that are evaluated in conjunction with the underlying bucket policy.
- Access grants are a simplified model for defining access permissions to data in Amazon S3 by prefix, bucket, or object. For information about S3 Access Grants, see Managing access with S3 Access Grants.

Principals for bucket policies

The `Principal` element specifies the user, account, service, or other entity that is either allowed or denied access to a resource. The following are examples of specifying `Principal`. For more information, see Principal in the *IAM User Guide*.

Grant permissions to an AWS account

To grant permissions to an AWS account, identify the account using the following format.

```
"AWS":"account-ARN"
```

The following are examples.

```
"Principal":{"AWS":"arn:aws:iam::AccountIDWithoutHyphens:root"}
```

```
"Principal":{"AWS":
["arn:aws:iam::AccountID1WithoutHyphens:root","arn:aws:iam::AccountID2WithoutHyphens:root"]}
```

> **Note**
>
> The examples above grant permissions to the root user, which delegates permissions to the account level. However, IAM policies are still required on the specific roles and users in the account.

Grant permissions to an IAM user

To grant permission to an IAM user within your account, you must provide an `"AWS":"user-ARN"` name-value pair.

```
"Principal":{"AWS":"arn:aws:iam::account-number-without-hyphens:user/username"}
```

For detailed examples that provide step-by-step instructions, see Example 1: Bucket owner granting its users bucket permissions and Example 3: Bucket owner granting permissions to objects it does not own.

> **Note**
>
> If an IAM identity is deleted after you update your bucket policy, the bucket policy will show a unique identifier in the principal element instead of an ARN. These unique IDs are never reused, so you can safely remove principals with unique identifiers from all of your policy statements. For more information about unique identifiers, see IAM identifiers in the *IAM User Guide*.

Grant anonymous permissions

> **Warning**
>
> Use caution when granting anonymous access to your Amazon S3 bucket. When you grant anonymous access, anyone in the world can access your bucket. We highly recommend that you never grant any kind of anonymous write access to your S3 bucket.

To grant permission to everyone, also referred as anonymous access, you set the wildcard (`"*"`) as the `Principal` value. For example, if you configure your bucket as a website, you want all the objects in the bucket to be publicly accessible.

```
"Principal":"*"
```

```
"Principal":{"AWS":"*"}
```

Using `"Principal": "*"` with an `Allow` effect in a resource-based policy allows anyone, even if they're not signed in to AWS, to access your resource.

Using `"Principal" : { "AWS" : "*" }` with an `Allow` effect in a resource-based policy allows any root user, IAM user, assumed-role session, or federated user in any account in the same partition to access your resource.

For anonymous users, these two methods are equivalent. For more information, see All principals in the *IAM User Guide*.

You cannot use a wildcard to match part of a principal name or ARN.

> **Important**
>
> In AWS access control policies, the Principals "*" and {"AWS": "*"} behave identically.

Restrict resource permissions

You can also use resource policy to restrict access to resources that would otherwise be available to IAM principals. Use a `Deny` statement to prevent access.

The following example blocks access if a secure transport protocol isn't used:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyBucketAccessIfSTPNotUsed",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
```

```
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      }
    }
  ]
}
```

Using `"Principal": "*"` so that this restriction applies to everyone is a best practice for this policy, instead of attempting to deny access only to specific accounts or principals using this method.

Require access through CloudFront URLs

You can require that your users access your Amazon S3 content only by using CloudFront URLs instead of Amazon S3 URLs. To do this, create a CloudFront origin access control (OAC). Then, change the permissions on your S3 data. In your bucket policy, you can set CloudFront as the Principal as follows:

```
"Principal":{"Service":"cloudfront.amazonaws.com"}
```

Use a `Condition` element in the policy to allow CloudFront to access the bucket only when the request is on behalf of the CloudFront distribution that contains the S3 origin.

```
       "Condition": {
          "StringEquals": {
             "AWS:SourceArn":
 "arn:aws:cloudfront::111122223333:distribution/CloudFront-distribution-ID"
          }
       }
```

For more information about requiring S3 access through CloudFront URLs, see Restricting access to an Amazon Simple Storage Service origin in the *Amazon CloudFront Developer Guide*. For more information about the security and privacy benefits of using Amazon CloudFront, see Configuring secure access and restricting access to content.

Resource-based policy examples for Amazon S3

- To view policy examples for Amazon S3 buckets, see Bucket policies for Amazon S3.
- To view policy examples for access points, see Configuring IAM policies for using access points.

Policy actions for Amazon S3

Supports policy actions: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Include actions in a policy to grant permissions to perform the associated operation.

The following shows different types of mapping relationship between S3 API operations and the required policy actions.

- One-to-one mapping with the same name. For example, to use the `PutBucketPolicy` API operation, the `s3:PutBucketPolicy` policy action is required.
- One-to-one mapping with different names. For example, to use the `ListObjectsV2` API operation, the `s3:ListBucket` policy action is required.
- One-to-many mapping. For example, to use the `HeadObject` API operation, the `s3:GetObject` is required. Also, when you use S3 Object Lock and want to get an object's Legal Hold status or retention settings, the corresponding `s3:GetObjectLegalHold` or `s3:GetObjectRetention` policy actions are also required before you can use the `HeadObject` API operation.
- Many-to-one mapping. For example, to use the `ListObjectsV2` or `HeadBucket` API operations, the `s3:ListBucket` policy action is required.

To see a list of Amazon S3 actions for use in policies, see Actions defined by Amazon S3 in the *Service Authorization Reference*. For a complete list of Amazon S3 API operations, see Amazon S3 API Actions in the *Amazon Simple Storage Service API Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Policy actions in Amazon S3 use the following prefix before the action:

```
s3
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "s3:action1",
      "s3:action2"
         ]
```

Bucket operations

Bucket operations are S3 API operations that operate on the bucket resource type. For example, `CreateBucket`, `ListObjectsV2`, and `PutBucketPolicy`. S3 policy actions for bucket operations require the `Resource` element in bucket policies or IAM identity-based policies to be the S3 bucket type Amazon Resource Name (ARN) identifier in the following example format.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
```

The following bucket policy grants the user *Akua* with account *12345678901* the `s3:ListBucket` permission to perform the ListObjectsV2 API operation and list objects in an S3 bucket.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to list objects in the bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
```

```
        }
    ]
}
```

Bucket operations in policies for access points for general purpose buckets

Permissions granted in an access point for general purpose buckets policy are effective only if the underlying bucket allows the same permissions. When you use S3 Access Points, you must delegate access control from the bucket to the access point or add the same permissions in the access point policies to the underlying bucket's policy. For more information, see Configuring IAM policies for using access points. In access point policies, S3 policy actions for bucket operations require you to use the access point ARN for the `Resource` element in the following format.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point"
```

The following access point policy grants the user *Akua* with account *12345678901* the `s3:ListBucket` permission to perform the ListObjectsV2 API operation through the S3 access point named *example-access-point*. This permission allows *Akua* to list the objects in the bucket that's associated with *example-access-point*.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAkuaToListObjectsInBucketThroughAccessPoint",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3:us-west-2:111122223333:accesspoint/example-
access-point"
        }
    ]
}
```

> **Note**
>
> Not all bucket operations are supported by access points for general purpose buckets. For more information, see Access points compatibility with S3 operations.

Bucket operations in policies for access points for directory buckets

Permissions granted in an access points for directory buckets policy are effective only if the underlying bucket allows the same permissions. When you use S3 Access Points, you must delegate access control from the bucket to the access point or add the same permissions in the access point policies to the underlying bucket's policy. For more information, see Configuring IAM policies for using access points for directory buckets. In access point policies, S3 policy actions for bucket operations require you to use the access point ARN for the `Resource` element in the following format.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point--usw2-az1--xa-s3"
```

The following access point policy grants the user *Akua* with account *12345678901* the `s3:ListBucket` permission to perform the ListObjectsV2 API operation through the access point named *example-access-point--usw2-az1--xa-s3*. This permission allows *Akua* to list the objects in the bucket that's associated with *example-access-point--usw2-az1--xa-s3*.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAkuaToListObjectsInTheBucketThroughAccessPoint",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Akua"
            },
            "Action": [
            "s3:ListBucket"
            ],
            "Resource": "arn:aws:s3express:us-east-1:111122223333:accesspoint/
example-access-point-usw2-az1-xa-s3"
```

```
        }
    ]
}
```

> **Note**
>
> Not all bucket operations are supported by access points for directory buckets. For more information, see Object operations for access points for directory buckets.

Object operations

Object operations are S3 API operations that act upon the object resource type. For example, `GetObject`, `PutObject`, and `DeleteObject`. S3 policy actions for object operations require the `Resource` element in policies to be the S3 object ARN in the following example formats.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
```

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/*"
```

> **Note**
>
> The object ARN must contain a forward slash after the bucket name, as seen in the previous examples.

The following bucket policy grants the user *Akua* with account *12345678901* the `s3:PutObject` permission. This permission allows *Akua* to use the PutObject API operation to upload objects to the S3 bucket named *amzn-s3-demo-bucket*.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to upload objects",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Akua"
            },
            "Action": [
            "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

Object operations in access point policies

When you use S3 Access Points to control access to object operations, you can use access point policies. When you use access point policies, S3 policy actions for object operations require you to use the access point ARN for the `Resource` element in the following format: `arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource`. For object operations that use access points, you must include the `/object/` value after the whole access point ARN in the `Resource` element. Here are some examples.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point/object/
*"
```

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point/
object/prefix/*"
```

The following access point policy grants the user *Akua* with account *12345678901* the `s3:GetObject` permission. This permission allows *Akua* to perform the GetObject API operation through the access point named *example-access-point* on all objects in the bucket that's associated with the access point.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to get objects through access point",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Akua"
            },
            "Action": [
            "s3:GetObject"
            ],
            "Resource": "arn:aws:s3:us-east-1:111122223333:accesspoint/example-
access-point/object/*"
        }
    ]
}
```

> **Note**
>
> Not all object operations are supported by access points. For more information, see Access points compatibility with S3 operations.

Object operations in policies for access points for directory buckets

When you use access points for directory buckets to control access to object operations, you can use access point policies. When you use access point policies, S3 policy actions for object operations require you to use the access point ARN for the `Resource` element in the following format: `arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource`. For object operations that use access points, you must include the `/object/` value after the whole access point ARN in the `Resource` element. Here are some examples.

```
"Resource": "arn:aws:s3express:us-west-2:123456789012:accesspoint/example-access-
point--usw2-az1--xa-s3/object/*"
```

```
"Resource": "arn:aws:s3express:us-west-2:123456789012:accesspoint/example-access-
point--usw2-az1--xa-s3/object/prefix/*"
```

The following access point policy grants the user *Akua* with account *12345678901* the `s3:GetObject` permission. This permission allows *Akua* to perform the GetObject API operation through the access point named *example-access-point--usw2-az1--xa-s3* on all objects in the bucket that's associated with the access point.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow Akua to get objects through access point",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::12345678901:user/Akua"
            },
            "Action": ["s3express:CreateSession", "s3:GetObject"],
            "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point--usw2-az1--xa-s3/object/*"
        }
    ]
}
```

> **Note**
>
> Not all object operations are supported by access points for directory buckets. For more information, see Object operations for access points for directory buckets.

Access point for general purpose bucket operations

Access point operations are S3 API operations that operate on the `accesspoint` resource type. For example, `CreateAccessPoint`, `DeleteAccessPoint`, and `GetAccessPointPolicy`. S3 policy actions for access point operations can only be used in IAM identity-based policies, not in bucket policies or access point policies. Access points operations require the `Resource` element to be the access point ARN in the following example format.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point"
```

The following IAM identity-based policy grants the `s3:GetAccessPointPolicy` permission to perform the GetAccessPointPolicy API operation on the S3 access point named *example-access-point*.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid":
 "GrantPermissionToRetrieveTheAccessPointPolicyOfAccessPointExampleAccessPoint",
            "Effect": "Allow",
            "Action": [
            "s3:GetAccessPointPolicy"
            ],
            "Resource": "arn:aws:s3:*:123456789012:accesspoint/example-access-
point"
        }
    ]
}
```

When you use Access Points, to control access to bucket operations, see Bucket operations in policies for access points for general purpose buckets; to control access to object operations, see Object operations in access point policies. For more information about how to configure access point policies, see Configuring IAM policies for using access points.

Access point for directory buckets operations

Access point for directory buckets operations are S3 API operations that operate on the `accesspoint` resource type. For example, `CreateAccessPoint`, `DeleteAccessPoint`, and `GetAccessPointPolicy`. S3 policy actions for access point operations can only be used in IAM identity-based policies, not in bucket policies or access point policies. Access points for directory buckets operations require the `Resource` element to be the access point ARN in the following example format.

```
"Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/example-access-point--usw2-
az1--xa-s3"
```

The following IAM identity-based policy grants the `s3express:GetAccessPointPolicy` permission to perform the GetAccessPointPolicy API operation on the access point named *example-access-point--usw2-az1--xa-s3*.

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid":
 "GrantPermissionToRetrieveTheAccessPointPolicyOfAccessPointExampleAccessPointUsw2Az1XaS3
            "Effect": "Allow",
            "Action": [
            "s3express:CreateSession","s3express:GetAccessPointPolicy"
            ],
            "Resource": "arn:aws:s3:*:111122223333:accesspoint/example-access-
point"
        }
    ]
}
```

The following IAM identity-based policy grants the `s3express:CreateAccessPoint` permission to create an access points for directory buckets.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Grant CreateAccessPoint.",
            "Principal": "*",
            "Action": [
                "s3express:CreateSession",
                "s3express:CreateAccessPoint"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

The following IAM identity-based policy grants the `s3express:PutAccessPointScope` permission to create access point scope for access points for directory buckets.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "Grant PutAccessPointScope",
            "Principal": "*",
            "Action": [
                "s3express:CreateSession",
                "s3express:CreateAccessPoint",
                "s3express:PutAccessPointScope"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

When you use access points for directory buckets to control access to bucket operations, see Bucket operations in policies for access points for directory buckets; to control access to object operations, see Object operations in policies for access points for directory buckets. For more information about how to configure access points for directory buckets policies, see Configuring IAM policies for using access points for directory buckets.

Object Lambda Access Point operations

With Amazon S3 Object Lambda, you can add your own code to Amazon S3 `GET`, `LIST`, and `HEAD` requests to modify and process data as it is returned to an application. You can make requests through an Object Lambda Access Point, which works the same as making requests through other access points. For more information, see Transforming objects with S3 Object Lambda.

For more information about how to configure policies for Object Lambda Access Point operations, see Configuring IAM policies for Object Lambda Access Points.

Multi-Region Access Point operations

A Multi-Region Access Point provides a global endpoint that applications can use to fulfill requests from S3 buckets that are located in multiple AWS Region. You can use a Multi-Region Access Point to build multi-Region applications with the same architecture that's used in a single Region, and then run those applications anywhere in the world. For more information, see Managing multi-Region traffic with Multi-Region Access Points.

For more information about how to configure policies for Multi-Region Access Point operations, see Multi-Region Access Point policy examples.

Batch job operations

(Batch Operations) job operations are S3 API operations that operate on the job resource type. For example, `DescribeJob` and `CreateJob`. S3 policy actions for job operations can only be used in IAM identity-based policies, not in bucket policies. Also, job operations require the `Resource` element in IAM identity-based policies to be the `job` ARN in the following example format.

```
"Resource": "arn:aws:s3:*:123456789012:job/*"
```

The following IAM identity-based policy grants the `s3:DescribeJob` permission to perform the DescribeJob API operation on the S3 Batch Operations job named *`example-job`*.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDescribingBatchOperationJob",
            "Effect": "Allow",
            "Action": [
            "s3:DescribeJob"
            ],
            "Resource": "arn:aws:s3:*:111122223333:job/example-job"
        }
    ]
}
```

S3 Storage Lens configuration operations

For more information about how to configure S3 Storage Lens configuration operations, see Setting Amazon S3 Storage Lens permissions.

Account operations

Account operations are S3 API operations that operate on the account level. For example, `GetPublicAccessBlock` (for account). Account isn't a resource type defined by Amazon S3. S3 policy actions for account operations can only be used in IAM identity-based policies, not in bucket policies. Also, account operations require the `Resource` element in IAM identity-based policies to be `"*"`.

The following IAM identity-based policy grants the `s3:GetAccountPublicAccessBlock` permission to perform the account-level GetPublicAccessBlock API operation and retrieve the account-level Public Access Block settings.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowRetrievingTheAccountLevelPublicAccessBlockSettings",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
       }
    ]
}
```

Policy examples for Amazon S3

- To view examples of Amazon S3 identity-based policies, see Identity-based policies for Amazon S3.
- To view examples of Amazon S3 resource-based policies, see Bucket policies for Amazon S3 and Configuring IAM policies for using access points.

Policy resources for Amazon S3

Supports policy resources: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. As a best practice, specify a resource using its Amazon Resource Name (ARN). For actions that don't

support resource-level permissions, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

Some Amazon S3 API actions support multiple resources. For example, `s3:GetObject` accesses *example-resource-1* and *example-resource-2*, so a principal must have permissions to access both resources. To specify multiple resources in a single statement, separate the ARNs with commas, as shown in the following example.

```
"Resource": [
      "example-resource-1",
      "example-resource-2"
```

Resources in Amazon S3 are buckets, objects, access points, or jobs. In a policy, use the Amazon Resource Name (ARN) of the bucket, object, access point, or job to identify the resource.

To see a complete list of Amazon S3 resource types and their ARNs, see Resources defined by Amazon S3 in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions defined by Amazon S3.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Wildcard characters in resource ARNs

You can use wildcard characters as part of the resource ARN. You can use the wildcard characters (* and ?) within any ARN segment (the parts separated by colons). An asterisk (*) represents any combination of zero or more characters, and a question mark (?) represents any single character. You can use multiple * or ? characters in each segment. However, a wildcard character can't span segments.

- The following ARN uses the * wildcard character in the `relative-ID` part of the ARN to identify all objects in the *amzn-s3-demo-bucket* bucket.

  ```
  arn:aws:s3:::amzn-s3-demo-bucket/*
  ```

- The following ARN uses * to indicate all S3 buckets and objects.

  ```
  arn:aws:s3:::*
  ```

- The following ARN uses both of the wildcard characters, * and ?, in the `relative-ID` part. This ARN identifies all objects in buckets such as *amzn-s3-demo-example1bucket*, *amzn-s3-demo-example2bucket*, *amzn-s3-demo-example3bucket*, and so on.

```
arn:aws:s3:::amzn-s3-demo-example?bucket/*
```

Policy variables for resource ARNs

You can use policy variables in Amazon S3 ARNs. At policy-evaluation time, these predefined variables are replaced by their corresponding values. Suppose that you organize your bucket as a collection of folders, with one folder for each of your users. The folder name is the same as the username. To grant users permission to their folders, you can specify a policy variable in the resource ARN:

```
arn:aws:s3:::bucket_name/developers/${aws:username}/
```

At runtime, when the policy is evaluated, the variable `${aws:username}` in the resource ARN is substituted with the username of the person who is making the request.

Policy examples for Amazon S3

- To view examples of Amazon S3 identity-based policies, see Identity-based policies for Amazon S3.
- To view examples of Amazon S3 resource-based policies, see Bucket policies for Amazon S3 and Configuring IAM policies for using access points.

Policy condition keys for Amazon S3

Supports service-specific policy condition keys: Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element specifies when statements execute based on defined criteria. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

Each Amazon S3 condition key maps to the same name request header allowed by the API on which the condition can be set. Amazon S3-specific condition keys dictate the behavior of the same name request headers. For example, the condition key `s3:VersionId` used to grant conditional permission for the
`s3:GetObjectVersion`
permission defines behavior of the `versionId` query parameter that you set in a GET Object request.

To see a list of Amazon S3 condition keys, see Condition keys for Amazon S3 in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions defined by Amazon S3.

Example: Restricting object uploads to objects with a specific storage class

Suppose that Account A, represented by account ID *123456789012*, owns a bucket. The Account A administrator wants to restrict *Dave*, a user in Account A, so that *Dave* can upload objects to the bucket only if the object is stored in the `STANDARD_IA` storage class. To restrict object uploads to a specific storage class, the Account A administrator can use the `s3:x-amz-storage-class` condition key, as shown in the following example bucket policy.

JSON

```
{
                "Version":"2012-10-17",
                "Statement": [
                  {
                    "Sid": "statement1",
                    "Effect": "Allow",
                    "Principal": {
                      "AWS": "arn:aws:iam::123456789012:user/Dave"
                    },
                    "Action": "s3:PutObject",
                    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                    "Condition": {
                      "StringEquals": {
                        "s3:x-amz-storage-class": [
                          "STANDARD_IA"
                        ]
                      }
                    }
                  }
```

```
            ]
        }
```

In the example, the `Condition` block specifies the `StringEquals` condition that is applied to the specified key-value pair, `"s3:x-amz-storage-class":["STANDARD_IA"]`. There is a set of predefined keys that you can use in expressing a condition. The example uses the `s3:x-amz-storage-class` condition key. This condition requires the user to include the `x-amz-storage-class` header with value `STANDARD_IA` in every `PutObject` request.

Policy examples for Amazon S3

- To view examples of Amazon S3 identity-based policies, see Identity-based policies for Amazon S3.
- To view examples of Amazon S3 resource-based policies, see Bucket policies for Amazon S3 and Configuring IAM policies for using access points.

ACLs in Amazon S3

Supports ACLs: Yes

In Amazon S3, access control lists (ACLs) control which AWS accounts have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

> **Important**
>
> A majority of modern use cases in Amazon S3 no longer require the use of ACLs.

For information about using ACLs to control access in Amazon S3, see Managing access with ACLs.

ABAC with Amazon S3

Supports ABAC (tags in policies): Partial

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes called tags. You can attach tags to IAM entities and AWS resources, then design ABAC policies to allow operations when the principal's tag matches the tag on the resource.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see Define permissions with ABAC authorization in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

For information about resources that support ABAC in Amazon S3, see Using tags for attribute-based access control (ABAC).

To view example identity-based policies for limiting access to S3 Batch Operations jobs based on tags, see Controlling permissions for Batch Operations using job tags.

ABAC and object tags

In ABAC policies, objects use `s3:` tags instead of `aws:` tags. To control access to objects based on object tags, you provide tag information in the Condition element of a policy using the following tags:

- `s3:ExistingObjectTag/tag-key`
- `s3:RequestObjectTagKeys`
- `s3:RequestObjectTag/tag-key`

For information about using object tags to control access, including example permission policies, see Tagging and access control policies.

Using temporary credentials with Amazon S3

Supports temporary credentials: Yes

Temporary credentials provide short-term access to AWS resources and are automatically created when you use federation or switch roles. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM and AWS services that work with IAM in the *IAM User Guide*.

Forward access sessions for Amazon S3

Supports forward access sessions (FAS): Yes

Forward access sessions (FAS) use the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. For policy details when making FAS requests, see Forward access sessions.

- FAS is used by Amazon S3 to make calls to AWS KMS to decrypt an object when SSE-KMS was used to encrypt it. For more information, see Using server-side encryption with AWS KMS keys (SSE-KMS).
- S3 Access Grants also uses FAS. After you create an access grant to your S3 data for a particular identity, the grantee requests a temporary credential from S3 Access Grants. S3 Access Grants obtains a temporary credential for the requester from AWS STS and vends the credential to the requester. For more information, see Request access to Amazon S3 data through S3 Access Grants.

Service roles for Amazon S3

Supports service roles: Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Create a role to delegate permissions to an AWS service in the *IAM User Guide*.

> **Warning**
>
> Changing the permissions for a service role might break Amazon S3 functionality. Edit service roles only when Amazon S3 provides guidance to do so.

Service-linked roles for Amazon S3

Supports service-linked roles: Partial

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

Amazon S3 supports service-linked roles for Amazon S3 Storage Lens. For details about creating or managing Amazon S3 service-linked roles, see Using service-linked roles for Amazon S3 Storage Lens.

Amazon S3 Service as a Principal

Service name in the policy	S3 feature	More information
`s3.amazonaws.com`	S3 Replication	Setting up live replication overview
`s3.amazonaws.com`	S3 event notifications	Amazon S3 Event Notifications
`s3.amazonaws.com`	S3 Inventory	Cataloging and analyzing your data with S3 Inventory
`access-grants.s3.amazonaws.com`	S3 Access Grants	Register a location
`batchoperations.s3.amazonaws.com`	S3 Batch Operations	Granting permissions for Batch Operations
`logging.s3.amazonaws.com`	S3 Server Access Logging	Enabling Amazon S3 server access logging
`storage-lens.s3.amazonaws.com`	S3 Storage Lens	Viewing Amazon S3 Storage Lens metrics using a data export

How Amazon S3 authorizes a request

When Amazon S3 receives a request—for example, a bucket or an object operation—it first verifies that the requester has the necessary permissions. Amazon S3 evaluates all the relevant access policies, user policies, and resource-based policies (bucket policy, bucket access control list (ACL), and object ACL) in deciding whether to authorize the request.

> **Note**
>
> If the Amazon S3 permission check fails to find valid permissions, an Access Denied (403 Forbidden)permission denied error is returned. For more information, see Troubleshoot Access Denied (403 Forbidden) errors in Amazon S3.

To determine whether the requester has permission to perform the specific operation, Amazon S3 does the following, in order, when it receives a request:

1. Converts all the relevant access policies (user policy, bucket policy, and ACLs) at run time into a set of policies for evaluation.
2. Evaluates the resulting set of policies in the following steps. In each step, Amazon S3 evaluates a subset of policies in a specific context, based on the context authority.
   a. **User context** – In the user context, the parent account to which the user belongs is the context authority.

      Amazon S3 evaluates a subset of policies owned by the parent account. This subset includes the user policy that the parent attaches to the user. If the parent also owns the resource in the request (bucket or object), Amazon S3 also evaluates the corresponding resource policies (bucket policy, bucket ACL, and object ACL) at the same time.

      A user must have permission from the parent account to perform the operation.

      This step applies only if the request is made by a user in an AWS account. If the request is made by using the root user credentials of an AWS account, Amazon S3 skips this step.

   b. **Bucket context** – In the bucket context, Amazon S3 evaluates policies owned by the AWS account that owns the bucket.

      If the request is for a bucket operation, the requester must have permission from the bucket owner. If the request is for an object, Amazon S3 evaluates all the policies owned by the

bucket owner to check if the bucket owner has not explicitly denied access to the object. If there is an explicit deny set, Amazon S3 does not authorize the request.

c. **Object context** – If the request is for an object, Amazon S3 evaluates the subset of policies owned by the object owner.

Following are some example scenarios that illustrate how Amazon S3 authorizes a request.

Example– Requester is an IAM principal

If the requester is an IAM principal, Amazon S3 must determine if the parent AWS account to which the principal belongs has granted the principal necessary permission to perform the operation. In addition, if the request is for a bucket operation, such as a request to list the bucket content, Amazon S3 must verify that the bucket owner has granted permission for the requester to perform the operation. To perform a specific operation on a resource, an IAM principal needs permission from both the parent AWS account to which it belongs and the AWS account that owns the resource.

Example– Requester is an IAM principal – If the request is for an operation on an object that the bucket owner doesn't own

If the request is for an operation on an object that the bucket owner doesn't own, in addition to making sure the requester has permissions from the object owner, Amazon S3 must also check the bucket policy to ensure the bucket owner has not set explicit deny on the object. A bucket owner (who pays the bill) can explicitly deny access to objects in the bucket regardless of who owns it. The bucket owner can also delete any object in the bucket.

By default, when another AWS account uploads an object to your S3 general purpose bucket, that account (the object writer) owns the object, has access to it, and can grant other users access to it through access control lists (ACLs). You can use Object Ownership to change this default behavior so that ACLs are disabled and you, as the bucket owner, automatically own every object in your general purpose bucket. As a result, access control for your data is based on policies, such as IAM user policies, S3 bucket policies, virtual private cloud (VPC) endpoint policies, and AWS Organizations service control policies (SCPs). For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

For more information about how Amazon S3 evaluates access policies to authorize or deny requests for bucket operations and object operations, see the following topics:

Topics

- How Amazon S3 authorizes a request for a bucket operation
- How Amazon S3 authorizes a request for an object operation

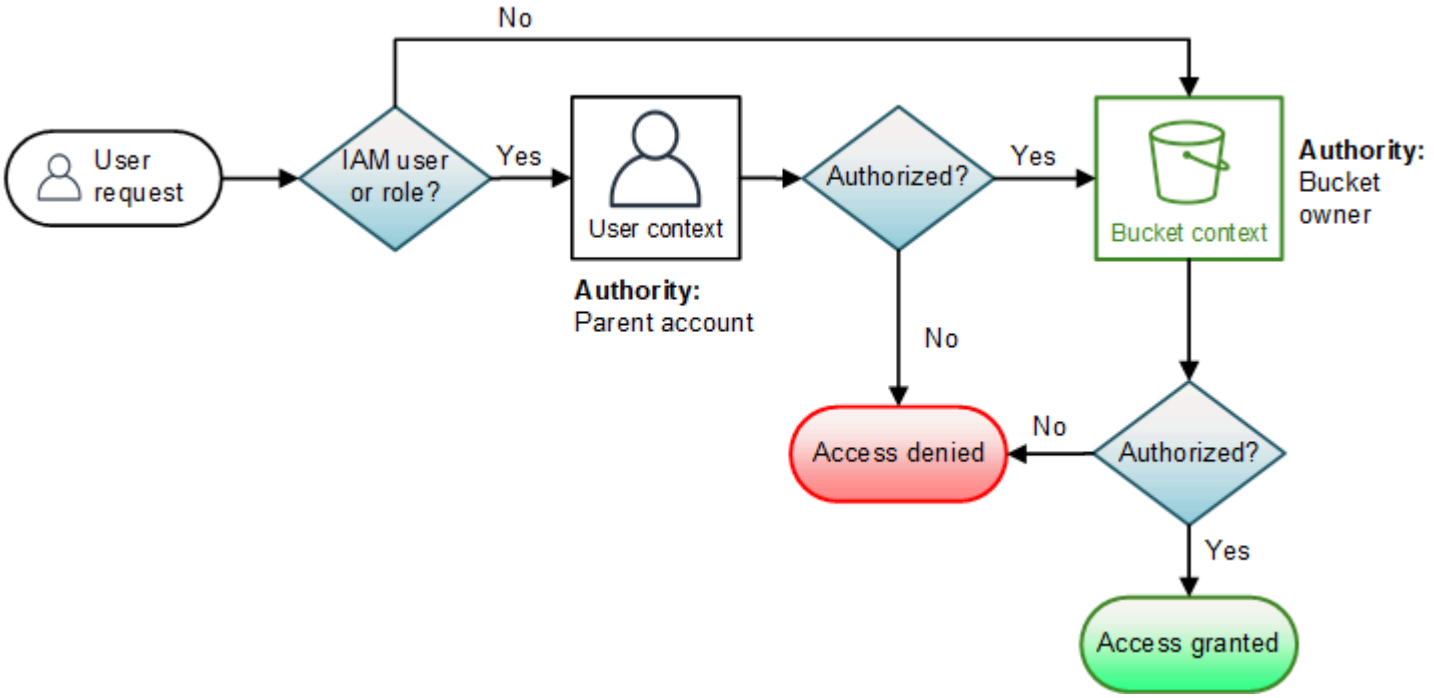
How Amazon S3 authorizes a request for a bucket operation

When Amazon S3 receives a request for a bucket operation, Amazon S3 converts all the relevant permissions into a set of policies to evaluate at run time. Relevant permissions include resource-based permissions (for example, bucket policies and bucket access control lists) and user policies if the request is from an IAM principal. Amazon S3 then evaluates the resulting set of policies in a series of steps according to a specific context—user context or bucket context:

1. **User context** – If the requester is an IAM principal, the principal must have permission from the parent AWS account to which it belongs. In this step, Amazon S3 evaluates a subset of policies owned by the parent account (also referred to as the context authority). This subset of policies includes the user policy that the parent account attaches to the principal. If the parent also owns the resource in the request (in this case, the bucket), Amazon S3 also evaluates the corresponding resource policies (bucket policy and bucket ACL) at the same time. Whenever a request for a bucket operation is made, the server access logs record the canonical ID of the requester. For more information, see Logging requests with server access logging.
2. **Bucket context** – The requester must have permissions from the bucket owner to perform a specific bucket operation. In this step, Amazon S3 evaluates a subset of policies owned by the AWS account that owns the bucket.

   The bucket owner can grant permission by using a bucket policy or bucket ACL. If the AWS account that owns the bucket is also the parent account of an IAM principal, then it can configure bucket permissions in a user policy.

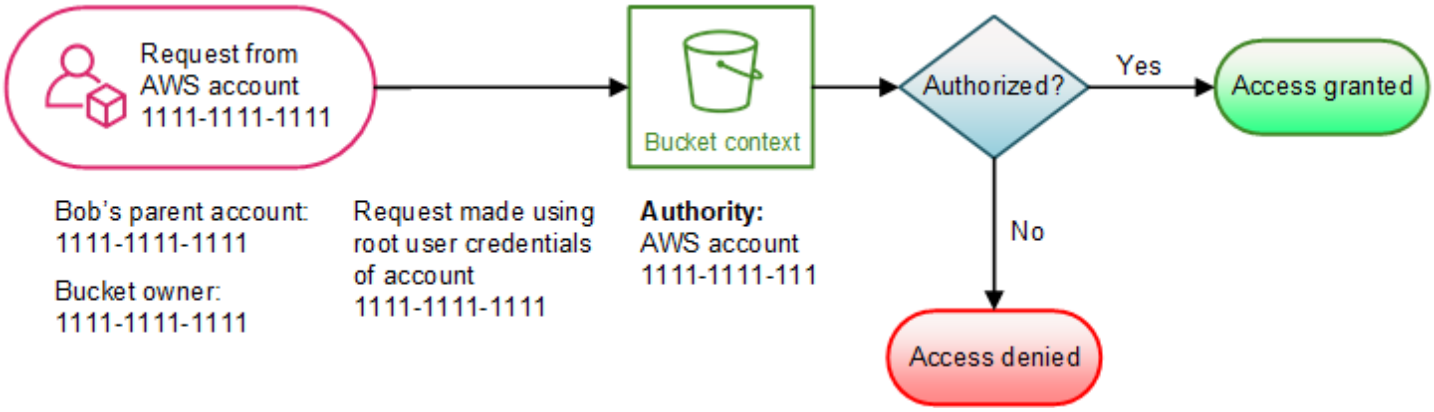
Context-based evaluation for bucket operation.



The following examples illustrate the evaluation logic.

Example 1: Bucket operation requested by bucket owner

In this example, the bucket owner sends a request for a bucket operation by using the root credentials of the AWS account.

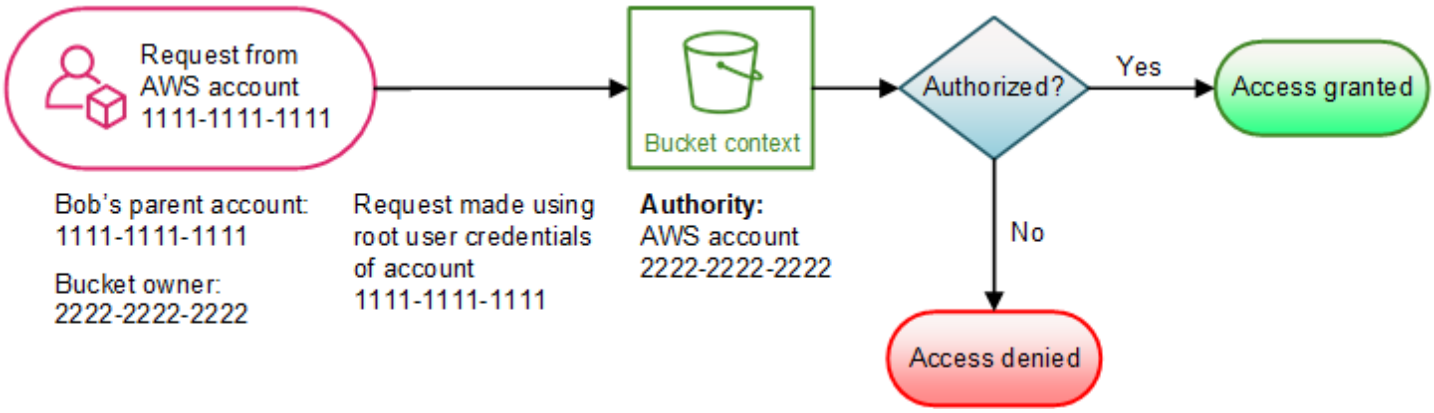


Amazon S3 performs the context evaluation as follows:

1. Because the request is made by using the root user credentials of an AWS account, the user context is not evaluated.
2. In the bucket context, Amazon S3 reviews the bucket policy to determine if the requester has permission to perform the operation. Amazon S3 authorizes the request.

Example 2: Bucket operation requested by an AWS account that is not the bucket owner

In this example, a request is made by using the root user credentials of AWS account 1111-1111-1111 for a bucket operation owned by AWS account 2222-2222-2222. No IAM users are involved in this request.

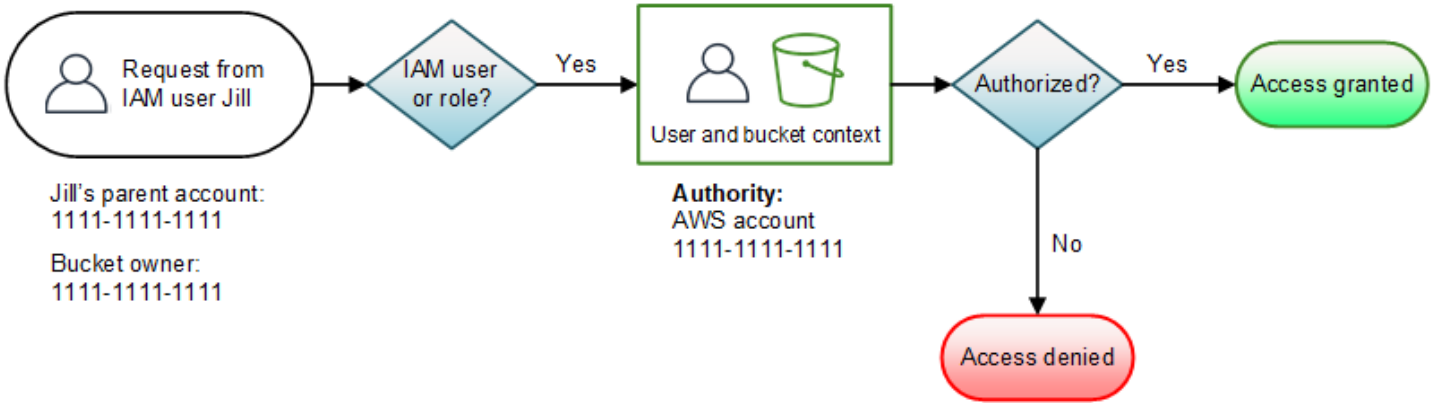


In this example, Amazon S3 evaluates the context as follows:

1. Because the request is made by using the root user credentials of an AWS account, the user context is not evaluated.
2. In the bucket context, Amazon S3 examines the bucket policy. If the bucket owner (AWS account 2222-2222-2222) has not authorized AWS account 1111-1111-1111 to perform the requested operation, Amazon S3 denies the request. Otherwise, Amazon S3 grants the request and performs the operation.

Example 3: Bucket operation requested by an IAM principal whose parent AWS account is also the bucket owner

In the example, the request is sent by Jill, an IAM user in AWS account 1111-1111-1111, which also owns the bucket.



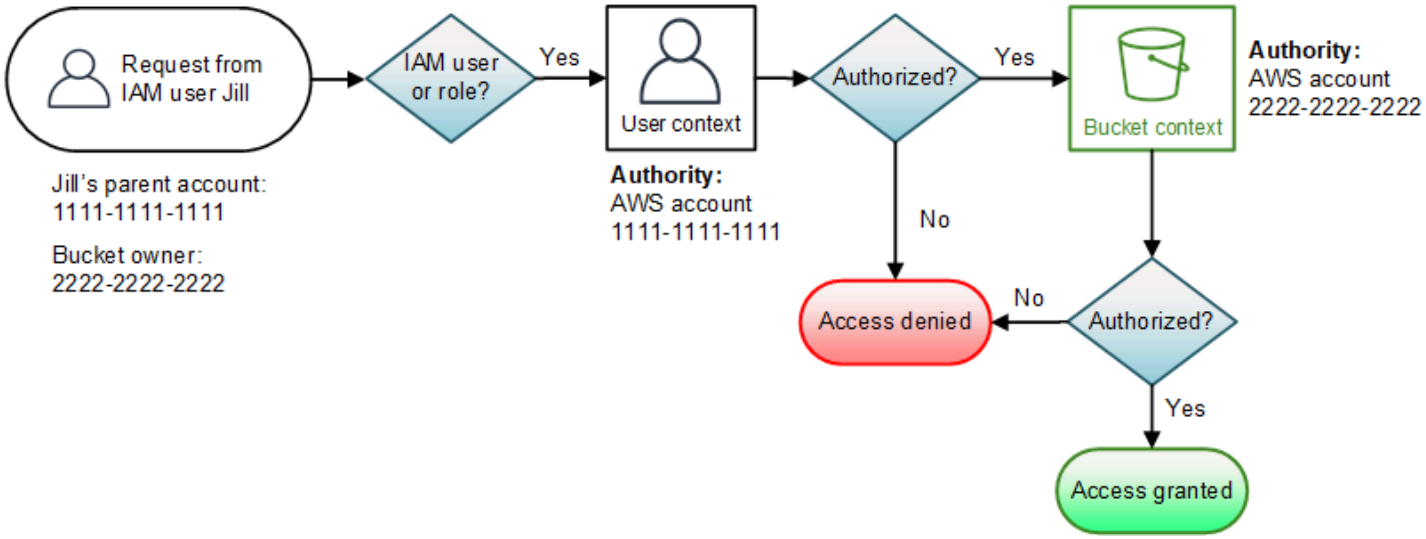
Amazon S3 performs the following context evaluation:

1. Because the request is from an IAM principal, in the user context, Amazon S3 evaluates all policies that belong to the parent AWS account to determine if Jill has permission to perform the operation.

   In this example, parent AWS account 1111-1111-1111, to which the principal belongs, is also the bucket owner. As a result, in addition to the user policy, Amazon S3 also evaluates the bucket policy and bucket ACL in the same context because they belong to the same account.
2. Because Amazon S3 evaluated the bucket policy and bucket ACL as part of the user context, it does not evaluate the bucket context.

Example 4: Bucket operation requested by an IAM principal whose parent AWS account is not the bucket owner

In this example, the request is sent by Jill, an IAM user whose parent AWS account is 1111-1111-1111, but the bucket is owned by another AWS account, 2222-2222-2222.



Jill will need permissions from both the parent AWS account and the bucket owner. Amazon S3 evaluates the context as follows:

1. Because the request is from an IAM principal, Amazon S3 evaluates the user context by reviewing the policies authored by the account to verify that Jill has the necessary permissions. If Jill has permission, then Amazon S3 moves on to evaluate the bucket context. If Jill doesn't have permission, it denies the request.

2. In the bucket context, Amazon S3 verifies that bucket owner 2222-2222-2222 has granted Jill (or her parent AWS account) permission to perform the requested operation. If she has that permission, Amazon S3 grants the request and performs the operation. Otherwise, Amazon S3 denies the request.

How Amazon S3 authorizes a request for an object operation

When Amazon S3 receives a request for an object operation, it converts all the relevant permissions — resource-based permissions (object access control list (ACL), bucket policy, bucket ACL) and IAM user policies—into a set of policies to be evaluated at run time. It then evaluates the resulting set of policies in a series of steps. In each step, it evaluates a subset of policies in three specific contexts—user context, bucket context, and object context:

1. **User context** – If the requester is an IAM principal, the principal must have permission from the parent AWS account to which it belongs. In this step, Amazon S3 evaluates a subset of policies owned by the parent account (also referred as the context authority). This subset of policies includes the user policy that the parent attaches to the principal. If the parent also owns the resource in the request (bucket or object), Amazon S3 evaluates the corresponding resource policies (bucket policy, bucket ACL, and object ACL) at the same time.

   > **Note**
   >
   > If the parent AWS account owns the resource (bucket or object), it can grant resource permissions to its IAM principal by using either the user policy or the resource policy.

2. **Bucket context** – In this context, Amazon S3 evaluates policies owned by the AWS account that owns the bucket.

   If the AWS account that owns the object in the request is not same as the bucket owner, Amazon S3 checks the policies if the bucket owner has explicitly denied access to the object. If there is an explicit deny set on the object, Amazon S3 does not authorize the request.

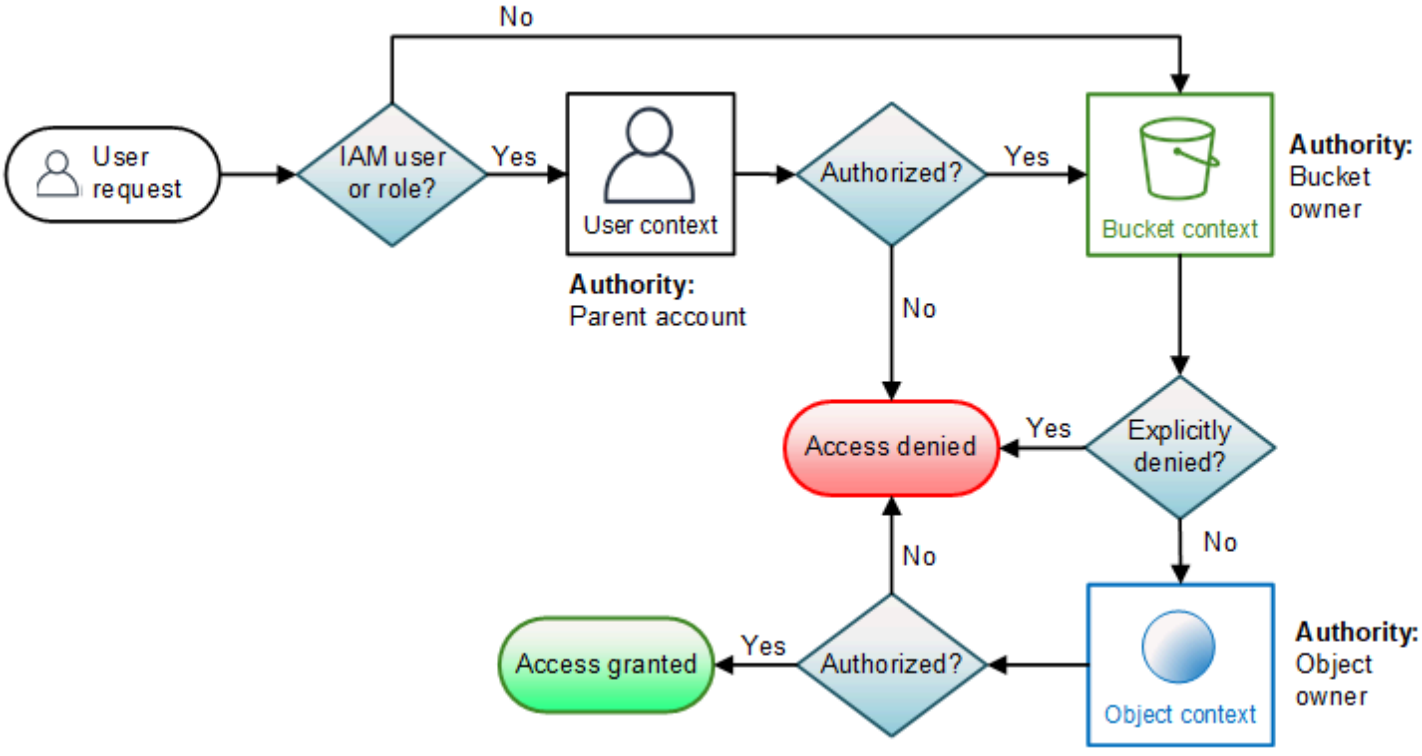
3. **Object context** – The requester must have permissions from the object owner to perform a specific object operation. In this step, Amazon S3 evaluates the object ACL.

   > **Note**
   >
   > If bucket and object owners are the same, access to the object can be granted in the bucket policy, which is evaluated at the bucket context. If the owners are different, the

object owners must use an object ACL to grant permissions. If the AWS account that owns the object is also the parent account to which the IAM principal belongs, it can configure object permissions in a user policy, which is evaluated at the user context. For more information about using these access policy alternatives, see Walkthroughs that use policies to manage access to your Amazon S3 resources.

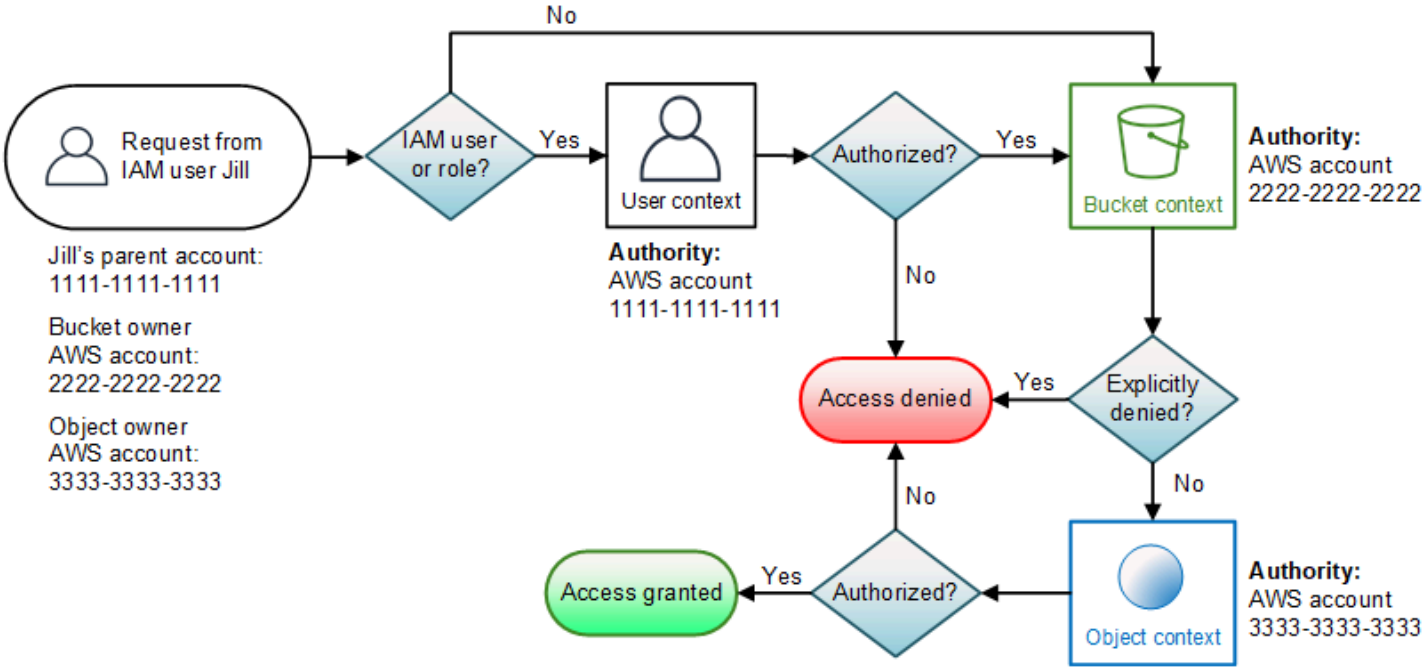
If you as the bucket owner want to own all the objects in your bucket and use bucket policies or policies based on IAMto manage access to these objects, you can apply the bucket owner enforced setting for Object Ownership. With this setting, you as the bucket owner automatically own and have full control over every object in your bucket. Bucket and object ACLs can't be edited and are no longer considered for access. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

The following is an illustration of the context-based evaluation for an object operation.



Example of an object operation request

In this example, IAM user Jill, whose parent AWS account is 1111-1111-1111, sends an object operation request (for example, `GetObject`) for an object owned by AWS account 3333-3333-3333 in a bucket owned by AWS account 2222-2222-2222.



Jill will need permission from the parent AWS account, the bucket owner, and the object owner. Amazon S3 evaluates the context as follows:

1. Because the request is from an IAM principal, Amazon S3 evaluates the user context to verify that the parent AWS account 1111-1111-1111 has given Jill permission to perform the requested operation. If she has that permission, Amazon S3 evaluates the bucket context. Otherwise, Amazon S3 denies the request.
2. In the bucket context, the bucket owner, AWS account 2222-2222-2222, is the context authority. Amazon S3 evaluates the bucket policy to determine if the bucket owner has explicitly denied Jill access to the object.
3. In the object context, the context authority is AWS account 3333-3333-3333, the object owner. Amazon S3 evaluates the object ACL to determine if Jill has permission to access the object. If she does, Amazon S3 authorizes the request.

Required permissions for Amazon S3 API operations

> **Note**
>
> This page is about Amazon S3 policy actions for general purpose buckets. To learn more about Amazon S3 policy actions for directory buckets, see Actions for directory buckets.

To perform an S3 API operation, you must have the right permissions. This page maps S3 API operations to the required permissions. To grant permissions to perform an S3 API operation, you must compose a valid policy (such as an S3 bucket policy or IAM identity-based policy), and specify corresponding actions in the `Action` element of the policy. These actions are called policy actions. Not every S3 API operation is represented by a single permission (a single policy action), and some permissions (some policy actions) are required for many different API operations.

When you compose policies, you must specify the `Resource` element based on the correct resource type required by the corresponding Amazon S3 policy actions. This page categorizes permissions to S3 API operations by the resource types. For more information about the resource types, see Resource types defined by Amazon S3 in the *Service Authorization Reference*. For a full list of Amazon S3 policy actions, resources, and condition keys for use in policies, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*. For a complete list of Amazon S3 API operations, see Amazon S3 API Actions in the *Amazon Simple Storage Service API Reference*.

For more information on how to address the HTTP `403 Forbidden` errors in S3, see Troubleshoot access denied (403 Forbidden) errors in Amazon S3. For more information on the IAM features to use with S3, see How Amazon S3 works with IAM. For more information on S3 security best practices, see Security best practices for Amazon S3.

Topics

- Bucket operations and permissions
- Object operations and permissions
- Access point for general purpose buckets operations and permissions
- Object Lambda Access Point operations and permissions
- Multi-Region Access Point operations and permissions
- Batch job operations and permissions
- S3 Storage Lens configuration operations and permissions
- S3 Storage Lens groups operations and permissions
- S3 Access Grants instance operations and permissions
- S3 Access Grants location operations and permissions
- S3 Access Grants grant operations and permissions
- Account operations and permissions

Bucket operations and permissions

Bucket operations are S3 API operations that operate on the bucket resource type. You must specify S3 policy actions for bucket operations in bucket policies or IAM identity-based policies.

In the policies, the `Resource` element must be the bucket Amazon Resource Name (ARN). For more information about the `Resource` element format and example policies, see Bucket operations.

Note

To grant permissions to bucket operations in access point policies, note the following:

- Permissions granted for bucket operations in an access point policy are effective only if the underlying bucket allows the same permissions. When you use an access point, you must delegate access control from the bucket to the access point or add the same permissions in the access point policy to the underlying bucket's policy.
- In access point policies that grant permissions to bucket operations, the `Resource` element must be the `accesspoint` ARN. For more information about the `Resource` element format and example policies, see Bucket operations in policies for access points for general purpose buckets. For more information about access point policies, see Configuring IAM policies for using access points.
- Not all bucket operations are supported by access points. For more information, see Access points compatibility with S3 operations.

The following is the mapping of bucket operations and required policy actions.

API operations	Policy actions	Description of policy actions
CreateBucket	(Required) `s3:CreateBucket`	Required to create a new s3 bucket.
	(Conditionally required) `s3:PutBucketAcl`	Required if you want to use access control list (ACL) to specify permissions on a bucket when you make a `CreateBucket` request.

API operations	Policy actions	Description of policy actions
	(Conditionally required) `s3:PutBucketObjectLockConfiguration` , `s3:PutBucketVersioning`	Required if you want to enable Object Lock when you create a bucket.
	(Conditionally required) `s3:PutBucketOwnershipControls`	Required if you want to specify S3 Object Ownership when you create a bucket.

API operations	Policy actions	Description of policy actions
CreateBucketMetadataConfiguration (V2 API operation. The IAM policy action name is the same for the V1 and V2 API operations.)	(Required) `s3:CreateBucketMetadataTableConfiguration`, `s3tables:CreateTableBucket`, `s3tables:CreateNamespace`, `s3tables:CreateTable`, `s3tables:GetTable`, `s3tables:PutTablePolicy`, `s3tables:PutTableEncryption`, `kms:DescribeKey`	Required to create a metadata table configuration on a general purpose bucket. To create your AWS managed table bucket and the metadata tables that are specified in your metadata table configuration, you must have the specified `s3tables` permissions. If you want to encrypt your metadata tables with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), you need additional permissions in your KMS key policy. For more information, see the section called "Permissions for metadata tables". If you also want to integrate your AWS managed table bucket with AWS analytics services so that you can query your metadata table, you need additional permissions. For more information, see Integrating Amazon S3 Tables with AWS analytics services.

API operations	Policy actions	Description of policy actions
CreateBucketMetadataTableConfiguration (V1 API operation)	(Required) `s3:CreateBucketMetadataTableConfiguration` , `s3tables:CreateNamespace` , `s3tables:CreateTable` , `s3tables:GetTable` , `s3tables:PutTablePolicy`	Required to create a metadata table configuration on a general purpose bucket. To create the metadata table in the table bucket that's specified in your metadata table configuration, you must have the specified `s3tables` permissions. If you want to encrypt your metadata tables with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), you need additional permissions. For more information, see the section called “Permissions for metadata tables”. If you also want to integrate your table bucket with AWS analytics services so that you can query your metadata table, you need additional permissions. For more information, see Integrating Amazon S3 Tables with AWS analytics services.
DeleteBucket	(Required) `s3:DeleteBucket`	Required to delete an S3 bucket.

API operations	Policy actions	Description of policy actions
DeleteBucketAnalyticsConfiguration	(Required) `s3:PutAnalyticsConfiguration`	Required to delete an S3 analytics configuration from an S3 bucket.
DeleteBucketCors	(Required) `s3:PutBucketCORS`	Required to delete the cross-origin resource sharing (CORS) configuration for an bucket.
DeleteBucketEncryption	(Required) `s3:PutEncryptionConfiguration`	Required to reset the default encryption configuration for an S3 bucket as server-side encryption with Amazon S3 managed keys (SSE-S3).
DeleteBucketIntelligentTieringConfiguration	(Required) `s3:PutIntelligentTieringConfiguration`	Required to delete the existing S3 Intelligent-Tiering configuration from an S3 bucket.
DeleteBucketInventoryConfiguration	(Required) `s3:PutInventoryConfiguration`	Required to delete an S3 Inventory configuration from an S3 bucket.
DeleteBucketLifecycle	(Required) `s3:PutLifecycleConfiguration`	Required to delete the S3 Lifecycle configuration for an S3 bucket.
DeleteBucketMetadataConfiguration (V2 API operation. The IAM policy action name is the same for the V1 and V2 API operations.)	(Required) `s3:DeleteBucketMetadataTableConfiguration`	Required to delete a metadata table configuration from a general purpose bucket.

API operations	Policy actions	Description of policy actions
DeleteBucketMetadataTableConfiguration (V1 API operation)	(Required) `s3:DeleteBucketMetadataTableConfiguration`	Required to delete a metadata table configuration from a general purpose bucket.
DeleteBucketMetricsConfiguration	(Required) `s3:PutMetricsConfiguration`	Required to delete a metrics configuration for the Amazon CloudWatch request metrics from an S3 bucket.
DeleteBucketOwnershipControls	(Required) `s3:PutBucketOwnershipControls`	Required to remove the Object Ownership setting for an S3 bucket. After removal, the Object Ownership setting becomes `Object writer`.
DeleteBucketPolicy	(Required) `s3:DeleteBucketPolicy`	Required to delete the policy of an S3 bucket.
DeleteBucketReplication	(Required) `s3:PutReplicationConfiguration`	Required to delete the replication configuration of an S3 bucket.
DeleteBucketTagging	(Required) `s3:PutBucketTagging`	Required to delete tags from an S3 bucket.
DeleteBucketWebsite	(Required) `s3:DeleteBucketWebsite`	Required to remove the website configuration for an S3 bucket.
DeletePublicAccessBlock (Bucket-level)	(Required) `s3:PutBucketPublicAccessBlock`	Required to remove the block public access configuration for an S3 bucket.

API operations	Policy actions	Description of policy actions
GetBucketAccelerateConfiguration	(Required) `s3:GetAccelerateConfiguration`	Required to use the accelerate subresource to return the Amazon S3 Transfer Acceleration state of a bucket, which is either Enabled or Suspended.
GetBucketAcl	(Required) `s3:GetBucketAcl`	Required to return the access control list (ACL) of an S3 bucket.
GetBucketAnalyticsConfiguration	(Required) `s3:GetAnalyticsConfiguration`	Required to return an analytics configuration that's identified by the analytics configuration ID from an S3 bucket.
GetBucketCors	(Required) `s3:GetBucketCORS`	Required to return the cross-origin resource sharing (CORS) configuration for an S3 bucket.
GetBucketEncryption	(Required) `s3:GetEncryptionConfiguration`	Required to return the default encryption configuration for an S3 bucket.
GetBucketIntelligentTieringConfiguration	(Required) `s3:GetIntelligentTieringConfiguration`	Required to get the S3 Intelligent-Tiering configuration of an S3 bucket.
GetBucketInventoryConfiguration	(Required) `s3:GetInventoryConfiguration`	Required to return an inventory configuration that's identified by the inventory configuration ID from the bucket.

API operations	Policy actions	Description of policy actions
GetBucketLifecycle	(Required) `s3:GetLifecycleConfiguration`	Required to return the S3 Lifecycle configuration of the bucket.
GetBucketLocation	(Required) `s3:GetBucketLocation`	Required to return the AWS Region that an S3 bucket resides in.
GetBucketLogging	(Required) `s3:GetBucketLogging`	Required to return the logging status of an S3 bucket and the permissions that users have to view and modify that status.
GetBucketMetadataConfiguration (V2 API operation. The IAM policy action name is the same for the V1 and V2 API operations.)	(Required) `s3:GetBucketMetadataTableConfiguration`	Required to retrieve a metadata table configuration for a general purpose bucket.
GetBucketMetadataTableConfiguration (V1 API operation)	(Required) `s3:GetBucketMetadataTableConfiguration`	Required to retrieve a metadata table configuration for a general purpose bucket.
GetBucketMetricsConfiguration	(Required) `s3:GetMetricsConfiguration`	Required to get a metrics configuration that's specified by the metrics configuration ID from the bucket.
GetBucketNotificationConfiguration	(Required) `s3:GetBucketNotification`	Required to return the notification configuration of an S3 bucket.

API operations	Policy actions	Description of policy actions
GetBucketOwnershipControls	(Required) `s3:GetBucketOwnershipControls`	Required to retrieve the Object Ownership setting for an S3 bucket.
GetBucketPolicy	(Required) `s3:GetBucketPolicy`	Required to return the policy of an S3 bucket.
GetBucketPolicyStatus	(Required) `s3:GetBucketPolicyStatus`	Required to retrieve the policy status for an S3 bucket, indicating whether the bucket is public.
GetBucketReplication	(Required) `s3:GetReplicationConfiguration`	Required to return the replication configuration of an S3 bucket.
GetBucketRequestPayment	(Required) `s3:GetBucketRequestPayment`	Required to return the request payment configuration for an S3 bucket.
GetBucketVersioning	(Required) `s3:GetBucketVersioning`	Required to return the versioning state of an S3 bucket.
GetBucketTagging	(Required) `s3:GetBucketTagging`	Required to return the tag set that's associated with an S3 bucket.
GetBucketWebsite	(Required) `s3:GetBucketWebsite`	Required to return the website configuration for an S3 bucket.
GetObjectLockConfiguration	(Required) `s3:GetBucketObjectLockConfiguration`	Required to get the Object Lock configuration for an S3 bucket.

API operations	Policy actions	Description of policy actions
GetPublicAccessBlock (Bucket-level)	(Required) `s3:GetBucketPublicAccessBlock`	Required to retrieve the block public access configuration for an S3 bucket.
HeadBucket	(Required) `s3:ListBucket`	Required to determine if a bucket exists and if you have permission to access it.
ListBucketAnalyticsConfigurations	(Required) `s3:GetAnalyticsConfiguration`	Required to list the analytics configurations for an S3 bucket.
ListBucketIntelligentTieringConfigurations	(Required) `s3:GetIntelligentTieringConfiguration`	Required to list the S3 Intelligent-Tiering configurations of an S3 bucket.
ListBucketInventoryConfigurations	(Required) `s3:GetInventoryConfiguration`	Required to return a list of inventory configurations for an S3 bucket.
ListBucketMetricsConfigurations	(Required) `s3:GetMetricsConfiguration`	Required to list the metrics configurations for an S3 bucket.
ListObjects	(Required) `s3:ListBucket`	Required to list some or all (up to 1,000) of the objects in an S3 bucket.
	(Conditionally required) `s3:GetObjectAcl`	Required if you want to display object owner information.
ListObjectsV2	(Required) `s3:ListBucket`	Required to list some or all (up to 1,000) of the objects in an S3 bucket.

API operations	Policy actions	Description of policy actions
	(Conditionally required) `s3:GetObjectAcl`	Required if you want to display object owner information.
ListObjectVersions	(Required) `s3:ListBucketVersions`	Required to get metadata about all the versions of objects in an S3 bucket.
PutBucketAccelerateConfiguration	(Required) `s3:PutAccelerateConfiguration`	Required to set the accelerate configuration of an existing bucket.
PutBucketAcl	(Required) `s3:PutBucketAcl`	Required to use access control lists (ACLs) to set the permissions on an existing bucket.
PutBucketAnalyticsConfiguration	(Required) `s3:PutAnalyticsConfiguration`	Required to set an analytics configuration for an S3 bucket.
PutBucketCors	(Required) `s3:PutBucketCORS`	Required to set the cross-origin resource sharing (CORS) configuration for an S3 bucket.
PutBucketEncryption	(Required) `s3:PutEncryptionConfiguration`	Required to configure the default encryption for an S3 bucket.
PutBucketIntelligentTieringConfiguration	(Required) `s3:PutIntelligentTieringConfiguration`	Required to put the S3 Intelligent-Tiering configuration to an S3 bucket.
PutBucketInventoryConfiguration	(Required) `s3:PutInventoryConfiguration`	Required to add an inventory configuration to an S3 bucket.

API operations	Policy actions	Description of policy actions
PutBucketLifecycle	(Required) `s3:PutLifecycleConfiguration`	Required to create a new S3 Lifecycle configuration or replace an existing lifecycle configuration for an S3 bucket.
PutBucketLogging	(Required) `s3:PutBucketLogging`	Required to set the logging parameters for an S3 bucket and specify permissions for who can view and modify the logging parameters.
PutBucketMetricsConfiguration	(Required) `s3:PutMetricsConfiguration`	Required to set or update a metrics configuration for the Amazon CloudWatch request metrics of an S3 bucket.
PutBucketNotificationConfiguration	(Required) `s3:PutBucketNotification`	Required to enable notifications of specified events for an S3 bucket.
PutBucketOwnershipControls	(Required) `s3:PutBucketOwnershipControls`	Required to create or modify the Object Ownership setting for an S3 bucket.
PutBucketPolicy	(Required) `s3:PutBucketPolicy`	Required to apply an S3 bucket policy to a bucket.
PutBucketReplication	(Required) `s3:PutReplicationConfiguration`	Required to create a new replication configuration or replace an existing one for an S3 bucket.
PutBucketRequestPayment	(Required) `s3:PutBucketRequestPayment`	Required to set the request payment configuration for a bucket.

API operations	Policy actions	Description of policy actions
PutBucketTagging	(Required) `s3:PutBucketTagging`	Required to add a set of tags to an S3 bucket.
PutBucketVersioning	(Required) `s3:PutBucketVersioning`	Required to set the versioning state of an S3 bucket.
PutBucketWebsite	(Required) `s3:PutBucketWebsite`	Required to configure a bucket as a website and set the configuration of the website.
PutObjectLockConfiguration	(Required) `s3:PutBucketObjectLockConfiguration`	Required to put Object Lock configuration on an S3 bucket.
PutPublicAccessBlock (Bucket-level)	(Required) `s3:PutBucketPublicAccessBlock`	Required to create or modify the block public access configuration for an S3 bucket.
UpdateBucketMetadataInventoryTableConfiguration	(Required) `s3:UpdateBucketMetadataInventoryTableConfiguration` , `s3tables:CreateTableBucket` , `s3tables:CreateNamespace` , `s3tables:CreateTable` , `s3tables:GetTable` , `s3tables:PutTablePolicy` , `s3tables:PutTableEncryption` , `kms:DescribeKey`	Required to enable or disable an inventory table for a metadata table configuration on a general purpose bucket. If you want to encrypt your inventory table with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), you need additional permissions in your KMS key policy. For more information, see the section called “Permissions for metadata tables”.

API operations	Policy actions	Description of policy actions
UpdateBucketMetadataJournalTableConfiguration	(Required) s3:UpdateBucketMetadataJournalTableConfiguration	Required to enable or disable journal table record expiration for a metadata table configuration on a general purpose bucket.

Object operations and permissions

Object operations are S3 API operations that operate on the object resource type. You must specify S3 policy actions for object operations in resource-based policies (such as bucket policies, access point policies, Multi-Region Access Point policies, VPC endpoint policies) or IAM identity-based policies.

In the policies, the `Resource` element must be the object ARN. For more information about the `Resource` element format and example policies, see Object operations.

Note

- AWS KMS policy actions (`kms:GenerateDataKey` and `kms:Decrypt`) are only applicable for the AWS KMS resource type and must be specified in IAM identity-based policies and AWS KMS resource-based policies (AWS KMS key policies). You can't specify AWS KMS policy actions in S3 resource-based policies, such as S3 bucket policies.
- When you use access points to control access to object operations, you can use access point policies. To grant permissions to object operations in access point policies, note the following:
  - In access point policies that grant permissions to object operations, the `Resource` element must be the ARNs for objects accessed through an access point. For more information about the `Resource` element format and example policies, see Object operations in access point policies.
  - Not all object operations are supported by access points. For more information, see Access points compatibility with S3 operations.
- Not all object operations are supported by Multi-Region Access Points. For more information, see Multi-Region Access Point compatibility with S3 operations.

The following is the mapping of object operations and required policy actions.

API operations	Policy actions	Description of policy actions
AbortMultipartUpload	(Required) `s3:AbortMultipartUpload`	Required to abort a multipart upload.
CompleteMultipartUpload	(Required) `s3:PutObject`	Required to complete a multipart upload.
	(Conditionally required) `kms:Decrypt`	Required if you want to complete a multipart upload for an AWS KMS customer managed key encrypted object.
CopyObject	For source object:	For source object:
	(Required) Either `s3:GetObject` or `s3:GetObjectVersion`	• `s3:GetObject` – Required if you want to copy an object from the source bucket without specifying `versionId` in the request. • `s3:GetObjectVersion` – Required if you want to copy a specific version of an object from the source bucket by specifying `versionId` in the request.
	(Conditionally required) `kms:Decrypt`	Required if you want to copy an AWS KMS customer managed key encrypted object from the source bucket.

API operations	Policy actions	Description of policy actions
	For destination object:	For destination object:
	(Required) `s3:PutObject`	Required to put the copied object in the destination bucket.
	(Conditionally required) `s3:PutObjectAcl`	Required if you want to put the copied object with the object access control list (ACL) to the destination bucket when you make a `CopyObject` request.
	(Conditionally required) `s3:PutObjectTagging`	Required if you want to put the copied object with object tagging to the destination bucket when you make a `CopyObject` request.
	(Conditionally required) `kms:GenerateDataKey`	Required if you want to encrypt the copied object with an AWS KMS customer managed key and put it to the destination bucket.
	(Conditionally required) `s3:PutObjectRetention`	Required if you want to set an Object Lock retention configuration for the new object.
	(Conditionally required) `s3:PutObjectLegalHold`	Required if you want to place an Object Lock legal hold on the new object.
CreateMultipartUpload	(Required) `s3:PutObject`	Required to create multipart upload.

API operations	Policy actions	Description of policy actions
	(Conditionally required) `s3:PutObjectAcl`	Required if you want to set the object access control list (ACL) permissions for the uploaded object.
	(Conditionally required) `s3:PutObjectTagging`	Required if you want to add object tagging(s) to the uploaded object.
	(Conditionally required) `kms:GenerateDataKey`	Required if you want to use an AWS KMS customer managed key to encrypt an object when you initiate a multipart upload.
	(Conditionally required) `s3:PutObjectRetention`	Required if you want to set an Object Lock retention configuration for the uploaded object.
	(Conditionally required) `s3:PutObjectLegalHold`	Required if you want to apply an Object Lock legal hold to the uploaded object.
DeleteObject	(Required) Either `s3:DeleteObject` or `s3:DeleteObjectVersion`	• `s3:DeleteObject` – Required if you want to remove an object without specifying `versionId` in the request. • `s3:DeleteObjectVersion` – Required if you want to remove a specific version of an object by specifying `versionId` in the request.

API operations	Policy actions	Description of policy actions
	(Conditionally required) `s3:BypassGovernanceRetention`	Required if you want to delete an object that's protected by governance mode for Object Lock retention.
DeleteObjects	(Required) Either `s3:DeleteObject` or `s3:DeleteObjectVersion`	• `s3:DeleteObject` – Required if you want to remove an object without specifying `versionId` in the request. • `s3:DeleteObjectVersion` – Required if you want to remove a specific version of an object by specifying `versionId` in the request.
	(Conditionally required) `s3:BypassGovernanceRetention`	Required if you want to delete objects that are protected by governance mode for Object Lock retention.

API operations	Policy actions	Description of policy actions
DeleteObjectTagging	(Required) Either `s3:DeleteObjectTagging` or `s3:DeleteObjectVersionTagging`	• `s3:DeleteObjectTagging` – Required if you want to remove the entire tag set of an object without specifying `versionId` in the request. • `s3:DeleteObjectVersionTagging` – Required if you want to delete tags of a specific object version by specifying `versionId` in the request.
GetObject	(Required) Either `s3:GetObject` or `s3:GetObjectVersion`	• `s3:GetObject` – Required if you want to get an object without specifying `versionId` in the request. • `s3:GetObjectVersion` – Required if you want to get a specific version of an object by specifying `versionId` in the request.
	(Conditionally required) `kms:Decrypt`	Required if you want to get and decrypt an AWS KMS customer managed key encrypted object.

API operations	Policy actions	Description of policy actions
	(Conditionally required) `s3:GetObjectTagging`	Required if you want to get the tag-set of an object when you make a `GetObject` request.
	(Conditionally required) `s3:GetObjectLegalHold`	Required if you want to get an object's current Object Lock legal hold status.
	(Conditionally required) `s3:GetObjectRetention`	Required if you want to retrieve the Object Lock retention settings for an object.
GetObjectAcl	(Required) Either `s3:GetObjectAcl` or `s3:GetObjectVersionAcl`	• `s3:GetObjectAcl` – Required if you want to get the access control list (ACL) of an object without specifying `versionId` in the request. • `s3:GetObjectVersionAcl` – Required if you want to get the access control list (ACL) of an object by specifying `versionId` in the request.

API operations	Policy actions	Description of policy actions
GetObjectAttributes	(Required) Either `s3:GetObject` or `s3:GetObjectVersion`	• `s3:GetObject` – Required if you want to retrieve attributes related to an object without specifying `versionId` in the request. • `s3:GetObjectVersion` – Required if you want to retrieve attributes related to a specific object version by specifying `versionId` in the request.
	(Conditionally required) `kms:Decrypt`	Required if you want to retrieve attributes related to an AWS KMS customer managed key encrypted object.
GetObjectLegalHold	(Required) `s3:GetObjectLegalHold`	Required to get an object's current Object Lock legal hold status.
GetObjectRetention	(Required) `s3:GetObjectRetention`	Required to retrieve the Object Lock retention settings for an object.

API operations	Policy actions	Description of policy actions
GetObjectTagging	(Required) Either `s3:GetObjectTagging` or `s3:GetObjectVersionTagging`	• `s3:GetObjectTagging` – Required if you want to get the tag set of an object without specifying `versionId` in the request. • `s3:GetObjectVersionTagging` – Required if you want to get the tags of a specific object version by specifying `versionId` in the request.
GetObjectTorrent	(Required) `s3:GetObject`	Required to return torrent files of an object.
HeadObject	(Required) `s3:GetObject`	Required to retrieve metadata from an object without returning the object itself.
	(Conditionally required) `s3:GetObjectLegalHold`	Required if you want to get an object's current Object Lock legal hold status.
	(Conditionally required) `s3:GetObjectRetention`	Required if you want to retrieve the Object Lock retention settings for an object.
ListMultipartUploads	(Required) `s3:ListBucketMultipartUploads`	Required to list in-progress multipart uploads in a bucket.
ListParts	(Required) `s3:ListMultipartUploadParts`	Required to list the parts that have been uploaded for a specific multipart upload.

API operations	Policy actions	Description of policy actions
	(Conditionally required) `kms:Decrypt`	Required if you want to list parts of an AWS KMS customer managed key encrypted multipart upload.
PutObject	(Required) `s3:PutObject`	Required to put an object.
	(Conditionally required) `s3:PutObjectAcl`	Required if you want to put the object access control list (ACL) when you make a `PutObject` request.
	(Conditionally required) `s3:PutObjectTagging`	Required if you want to put object tagging when you make a `PutObject` request.
	(Conditionally required) `kms:GenerateDataKey`	Required if you want to encrypt an object with an AWS KMS customer managed key.
	(Conditionally required) `s3:PutObjectRetention`	Required if you want to set an Object Lock retention configuration on an object.
	(Conditionally required) `s3:PutObjectLegalHold`	Required if you want to apply an Object Lock legal hold configuration to a specified object.

API operations	Policy actions	Description of policy actions
PutObjectAcl	(Required) Either `s3:PutObjectAcl` or `s3:PutObjectVersionAcl`	• `s3:PutObjectAcl` – Required if you want to set the access control list (ACL) permissions for a new or existing object without specifying `versionId` in the request. • `s3:PutObjectVersionAcl` – Required if you want to set the access control list (ACL) permissions for a new or existing object by specifying `versionId` in the request.
PutObjectLegalHold	(Required) `s3:PutObjectLegalHold`	Required to apply an Object Lock legal hold configuration to an object.
PutObjectRetention	(Required) `s3:PutObjectRetention`	Required to apply an Object Lock retention configuration to an object.
	(Conditionally required) `s3:BypassGovernanceRetention`	Required if you want to bypass the governance mode of an Object Lock retention configuration.

API operations	Policy actions	Description of policy actions
PutObjectTagging	(Required) Either `s3:PutObjectTagging` or `s3:PutObjectVersionTagging`	• `s3:PutObjectTagging` – Required if you want to set the supplied tag set to an object that already exists in a bucket without specifying `versionId` in the request. • `s3:PutObjectVersionTagging` – Required if you want to set the supplied tag set to an object that already exists in a bucket by specifying `versionId` in the request.
RestoreObject	(Required) `s3:RestoreObject`	Required to restore a copy of an archived object.
SelectObjectContent	(Required) `s3:GetObject`	Required to filter the contents of an S3 object based on a simple structured query language (SQL) statement.
	(Conditionally required) `kms:Decrypt`	Required if you want to filter the contents of an S3 object that's encrypted with an AWS KMS customer managed key.

API operations	Policy actions	Description of policy actions
UpdateObjectEncryption	(Required) `s3:UpdateObjectEncryption` , `kms:Encrypt` , `kms:Decrypt` , `kms:GenerateDataKey` , `kms:ReEncrypt*`	Required if you want to change encrypted objects between server-side encryption with Amazon S3 managed encryption (SSE-S3) and server-side encryption with AWS Key Management Service (AWS KMS) encryption keys (SSE-KMS). You can also use the `UpdateObjectEncryption` operation to apply S3 Bucket Keys to reduce AWS KMS request costs or change the customer-managed KMS key that's used to encrypt your data so that you can comply with custom key-rotation standards.

API operations	Policy actions	Description of policy actions
	(Conditionally required) `organizations:DescribeAccount`	If you're using AWS Organizations, to use the `UpdateObjectEncryption` operation with customer-managed KMS keys from other AWS accounts within your organization, you must have the `organizations:DescribeAccount` permission. **Note** You must also request the ability to use AWS KMS keys owned by other member accounts within your organization by contacting AWS Support.
UploadPart	(Required) `s3:PutObject`	Required to upload a part in a multipart upload.
	(Conditionally required) `kms:GenerateDataKey`	Required if you want to put an upload part and encrypt it with an AWS KMS customer managed key.
UploadPartCopy	For source object:	For source object:

API operations	Policy actions	Description of policy actions
	(Required) Either `s3:GetObject` or `s3:GetObjectVersion`	• `s3:GetObject` – Required if you want to copy an object from the source bucket without specifying `versionId` in the request. • `s3:GetObjectVersion` – Required if you want to copy a specific version of an object from the source bucket by specifying `versionId` in the request.
	(Conditionally required) `kms:Decrypt`	Required if you want to copy an AWS KMS customer managed key encrypted object from the source bucket.
	For destination part:	For destination part:
	(Required) `s3:PutObject`	Required to upload a multipart upload part to the destination bucket.
	(Conditionally required) `kms:GenerateDataKey`	Required if you want to encrypt a part with an AWS KMS customer managed key when you upload the part to the destination bucket.

Access point for general purpose buckets operations and permissions

Access point operations are S3 API operations that operate on the `accesspoint` resource type. You must specify S3 policy actions for access point operations in IAM identity-based policies, not in bucket policies or access point policies.

In the policies, the `Resource` element must be the `accesspoint` ARN. For more information about the `Resource` element format and example policies, see Access point for general purpose bucket operations.

Note

If you want to use access points to control access to bucket or object operations, note the following:

- For using access points to control access to bucket operations, see Bucket operations in policies for access points for general purpose buckets.
- For using access points to control access to object operations, see Object operations in access point policies.
- For more information about how to configure access point policies, see Configuring IAM policies for using access points.

The following is the mapping of access point operations and required policy actions.

API operations	Policy actions	Description of policy actions
CreateAccessPoint	(Required) `s3:CreateAccessPoint`	Required to create an access point that's associated with an S3 bucket.
DeleteAccessPoint	(Required) `s3:DeleteAccessPoint`	Required to delete an access point.
DeleteAccessPointPolicy	(Required) `s3:DeleteAccessPointPolicy`	Required to delete an access point policy.

API operations	Policy actions	Description of policy actions
GetAccessPointPolicy	(Required) `s3:GetAccessPointPolicy`	Required to retrieve an access point policy.
GetAccessPointPolicyStatus	(Required) `s3:GetAccessPointPolicyStatus`	Required to retrieve the information on whether the specified access point currently has a policy that allows public access.
PutAccessPointPolicy	(Required) `s3:PutAccessPointPolicy`	Required to put an access point policy.

Object Lambda Access Point operations and permissions

Object Lambda Access Point operations are S3 API operations that operate on the `objectlambdaaccesspoint` resource type. For more information about how to configure policies for Object Lambda Access Point operations, see Configuring IAM policies for Object Lambda Access Points.

The following is the mapping of Object Lambda Access Point operations and required policy actions.

API operations	Policy actions	Description of policy actions
CreateAccessPointForObjectLambda	(Required) `s3:CreateAccessPointForObjectLambda`	Required to create an Object Lambda Access Point.
DeleteAccessPointForObjectLambda	(Required) `s3:DeleteAccessPointForObjectLambda`	Required to delete a specified Object Lambda Access Point.
DeleteAccessPointPolicyForObjectLambda	(Required) `s3:DeleteAccessPointPolicyForObjectLambda`	Required to delete the policy on a specified Object Lambda Access Point.

API operations	Policy actions	Description of policy actions
GetAccessPointConfigurationForObjectLambda	(Required) `s3:GetAccessPointConfigurationForObjectLambda`	Required to retrieve the configuration of the Object Lambda Access Point.
GetAccessPointForObjectLambda	(Required) `s3:GetAccessPointForObjectLambda`	Required to retrieve information about the Object Lambda Access Point.
GetAccessPointPolicyForObjectLambda	(Required) `s3:GetAccessPointPolicyForObjectLambda`	Required to return the access point policy that's associated with the specified Object Lambda Access Point.
GetAccessPointPolicyStatusForObjectLambda	(Required) `s3:GetAccessPointPolicyStatusForObjectLambda`	Required to return the policy status for a specific Object Lambda Access Point policy.
PutAccessPointConfigurationForObjectLambda	(Required) `s3:PutAccessPointConfigurationForObjectLambda`	Required to set the configuration of the Object Lambda Access Point.
PutAccessPointPolicyForObjectLambda	(Required) `s3:PutAccessPointPolicyForObjectLambda`	Required to associate an access policy with a specified Object Lambda Access Point.

Multi-Region Access Point operations and permissions

Multi-Region Access Point operations are S3 API operations that operate on the `multiregionaccesspoint` resource type. For more information about how to configure policies for Multi-Region Access Point operations, see Multi-Region Access Point policy examples.

The following is the mapping of Multi-Region Access Point operations and required policy actions.

API operations	Policy actions	Description of policy actions
CreateMultiRegionAccessPoint	(Required) `s3:CreateMultiRegionAccessPoint`	Required to create a Multi-Region Access Point and associate it with S3 buckets.
DeleteMultiRegionAccessPoint	(Required) `s3:DeleteMultiRegionAccessPoint`	Required to delete a Multi-Region Access Point.
DescribeMultiRegionAccessPointOperation	(Required) `s3:DescribeMultiRegionAccessPointOperation`	Required to retrieve the status of an asynchronous request to manage a Multi-Region Access Point.
GetMultiRegionAccessPoint	(Required) `s3:GetMultiRegionAccessPoint`	Required to return configuration information about the specified Multi-Region Access Point.
GetMultiRegionAccessPointPolicy	(Required) `s3:GetMultiRegionAccessPointPolicy`	Required to return the access control policy of the specified Multi-Region Access Point.
GetMultiRegionAccessPointPolicyStatus	(Required) `s3:GetMultiRegionAccessPointPolicyStatus`	Required to return the policy status for a specific Multi-Region Access Point about whether the specified Multi-Region Access Point has an access control policy that allows public access.
GetMultiRegionAccessPointRoutes	(Required) `s3:GetMultiRegionAccessPointRoutes`	Required to return the routing configuration for a Multi-Region Access Point.

API operations	Policy actions	Description of policy actions
PutMultiRegionAccessPointPolicy	(Required) `s3:PutMultiRegionAccessPointPolicy`	Required to update the access control policy of the specified Multi-Region Access Point.
SubmitMultiRegionAccessPointRoutes	(Required) `s3:SubmitMultiRegionAccessPointRoutes`	Required to submit an updated route configuration for a Multi-Region Access Point.

Batch job operations and permissions

(Batch Operations) job operations are S3 API operations that operate on the `job` resource type. You must specify S3 policy actions for job operations in IAM identity-based policies, not in bucket policies.

In the policies, the `Resource` element must be the `job` ARN. For more information about the `Resource` element format and example policies, see Batch job operations.

The following is the mapping of batch job operations and required policy actions.

API operations	Policy actions	Description of policy actions
DeleteJobTagging	(Required) `s3:DeleteJobTagging`	Required to remove tags from an existing S3 Batch Operations job.
DescribeJob	(Required) `s3:DescribeJob`	Required to retrieve the configuration parameters and status for a Batch Operations job.
GetJobTagging	(Required) `s3:GetJobTagging`	Required to return the tag set of an existing S3 Batch Operations job.

API operations	Policy actions	Description of policy actions
PutJobTagging	(Required) `s3:PutJobTagging`	Required to put or replace tags on an existing S3 Batch Operations job.
UpdateJobPriority	(Required) `s3:UpdateJobPriority`	Required to update the priority of an existing job.
UpdateJobStatus	(Required) `s3:UpdateJobStatus`	Required to update the status for the specified job.

S3 Storage Lens configuration operations and permissions

S3 Storage Lens configuration operations are S3 API operations that operate on the `storagelensconfiguration` resource type. For more information about how to configure S3 Storage Lens configuration operations, see Setting Amazon S3 Storage Lens permissions.

The following is the mapping of S3 Storage Lens configuration operations and required policy actions.

API operations	Policy actions	Description of policy actions
DeleteStorageLensConfiguration	(Required) `s3:DeleteStorageLensConfiguration`	Required to delete the S3 Storage Lens configuration.
DeleteStorageLensConfigurationTagging	(Required) `s3:DeleteStorageLensConfigurationTagging`	Required to delete the S3 Storage Lens configuration tags.
GetStorageLensConfiguration	(Required) `s3:GetStorageLensConfiguration`	Required to get the S3 Storage Lens configuration.
GetStorageLensConfigurationTagging	(Required) `s3:GetStorageLensConfigurationTagging`	Required to get the tags of S3 Storage Lens configuration.

API operations	Policy actions	Description of policy actions
PutStorageLensConfigurationTagging	(Required) `s3:PutStorageLensConfigurationTagging`	Required to put or replace tags on an existing S3 Storage Lens configuration.

S3 Storage Lens groups operations and permissions

S3 Storage Lens groups operations are S3 API operations that operate on the `storagelensgroup` resource type. For more information about how to configure S3 Storage Lens groups permissions, see Storage Lens groups permissions.

The following is the mapping of S3 Storage Lens groups operations and required policy actions.

API operations	Policy actions	Description of policy actions
DeleteStorageLensGroup	(Required) `s3:DeleteStorageLensGroup`	Required to delete an existing S3 Storage Lens group.
GetStorageLensGroup	(Required) `s3:GetStorageLensGroup`	Required to retrieve the S3 Storage Lens group configuration details.
UpdateStorageLensGroup	(Required) `s3:UpdateStorageLensGroup`	Required to update the existing S3 Storage Lens group.
CreateStorageLensGroup	(Required) `s3:CreateStorageLensGroup`	Required to create a new Storage Lens group.
CreateStorageLensGroup, TagResource	(Required) `s3:CreateStorageLensGroup` , `s3:TagResource`	Required to create a new Storage Lens group with tags.
ListStorageLensGroups	(Required) `s3:ListStorageLensGroups`	Required to list all Storage Lens groups in your home Region.

API operations	Policy actions	Description of policy actions
ListTagsForResource	(Required) `s3:ListTagsForResource`	Required to list the tags that were added to your Storage Lens group.
TagResource	(Required) `s3:TagResource`	Required to add or update a Storage Lens group tag for an existing Storage Lens group.
UntagResource	(Required) `s3:UntagResource`	Required to delete a tag from a Storage Lens group.

S3 Access Grants instance operations and permissions

S3 Access Grants instance operations are S3 API operations that operate on the `accessgrantsinstance` resource type. An S3 Access Grants instance is a logical container for your access grants. For more information on working with S3 Access Grants instances, see Working with S3 Access Grants instances.

The following is the mapping of the S3 Access Grants instance configuration operations and required policy actions.

API operations	Policy actions	Description of policy actions
AssociateAccessGrantsIdentityCenter	(Required) `s3:AssociateAccessGrantsIdentityCenter`	Required to associate an AWS IAM Identity Center instance with your S3 Access Grants instance, thus enabling you to create access grants for users and groups in your corporate identity directory . You must also have the following permissions: `sso:CreateApplication` , `sso:PutApplicationGrant` , and

API operations	Policy actions	Description of policy actions
		`sso:PutApplicationAuthenticationMethod` .
CreateAccessGrantsInstance	(Required) `s3:CreateAccessGrantsInstance`	Required to create an S3 Access Grants instance (`accessgrantsinstance` resource) which is a container for your individual access grants. To associate an AWS IAM Identity Center instance with your S3 Access Grants instance, you must also have the `sso:DescribeInstance` , `sso:CreateApplication` , `sso:PutApplicationGrant` , and `sso:PutApplicationAuthenticationMethod` permissions.
DeleteAccessGrantsInstance	(Required) `s3:DeleteAccessGrantsInstance`	Required to delete an S3 Access Grants instance (`accessgrantsinstance` resource) from an AWS Region in your account.
DeleteAccessGrantsInstanceResourcePolicy	(Required) `s3:DeleteAccessGrantsInstanceResourcePolicy`	Required to delete a resource policy for your S3 Access Grants instance.

API operations	Policy actions	Description of policy actions
DissociateAccessGrantsIdentityCenter	(Required) `s3:DissociateAccessGrantsIdentityCenter`	Required to disassociate an AWS IAM Identity Center instance from your S3 Access Grants instance. You must also have the following permissions: `sso:DeleteApplication`
GetAccessGrantsInstance	(Required) `s3:GetAccessGrantsInstance`	Required to retrieve the S3 Access Grants instance for an AWS Region in your account.
GetAccessGrantsInstanceForPrefix	(Required) `s3:GetAccessGrantsInstanceForPrefix`	Required to retrieve the S3 Access Grants instance that contains a particular prefix.
GetAccessGrantsInstanceResourcePolicy	(Required) `s3:GetAccessGrantsInstanceResourcePolicy`	Required to return the resource policy of your S3 Access Grants instance.
ListAccessGrantsInstances	(Required) `s3:ListAccessGrantsInstances`	Required to return a list of the S3 Access Grants instances in your account.
PutAccessGrantsInstanceResourcePolicy	(Required) `s3:PutAccessGrantsInstanceResourcePolicy`	Required to update the resource policy of the S3 Access Grants instance.

S3 Access Grants location operations and permissions

S3 Access Grants location operations are S3 API operations that operate on the `accessgrantslocation` resource type. For more information on working with S3 Access Grants locations, see Working with S3 Access Grants locations.

The following is the mapping of the S3 Access Grants location configuration operations and required policy actions.

API operations	Policy actions	Description of policy actions
CreateAccessGrantsLocation	(Required) `s3:CreateAccessGrantsLocation`	Required to register a location in your S3 Access Grants instance (create an `accessgrantslocation` resource). You must also have the following permission for the specified IAM role: `iam:PassRole`
DeleteAccessGrantsLocation	(Required) `s3:DeleteAccessGrantsLocation`	Required to remove a registered location from your S3 Access Grants instance.
GetAccessGrantsLocation	(Required) `s3:GetAccessGrantsLocation`	Required to retrieve the details of a particular location registered in your S3 Access Grants instance.
ListAccessGrantsLocations	(Required) `s3:ListAccessGrantsLocations`	Required to return a list of the locations registered in your S3 Access Grants instance.
UpdateAccessGrantsLocation	(Required) `s3:UpdateAccessGrantsLocation`	Required to update the IAM role of a registered location in your S3 Access Grants instance.

S3 Access Grants grant operations and permissions

S3 Access Grants grant operations are S3 API operations that operate on the `accessgrant` resource type. For more information on working with individual grants using S3 Access Grants, see Working with grants in S3 Access Grants.

The following is the mapping of the S3 Access Grants grant configuration operations and required policy actions.

API operations	Policy actions	Description of policy actions
CreateAccessGrant	(Required) `s3:CreateAccessGrant`	Required to create an individual grant (`accessgrant` resource) for a user or group in your S3 Access Grants instance. You must also have the following permissions: For any directory identity — `sso:DescribeInstance` and `sso:DescribeApplication` For directory users — `identitystore:DescribeUser`
DeleteAccessGrant	(Required) `s3:DeleteAccessGrant`	Required to delete an individual access grant (`accessgrant` resource) from your S3 Access Grants instance.
GetAccessGrant	(Required) `s3:GetAccessGrant`	Required to get the details about an individual access grant in your S3 Access Grants instance.

API operations	Policy actions	Description of policy actions
ListAccessGrants	(Required) `s3:ListAccessGrants`	Required to return a list of individual access grants in your S3 Access Grants instance.
ListCallerAccessGrants	(Required) `s3:ListCallerAccessGrants`	Required to list the access grants that grant the caller access to Amazon S3 data through S3 Access Grants.

Account operations and permissions

Account operations are S3 API operations that operate on the account level. Account isn't a resource type defined by Amazon S3. You must specify S3 policy actions for account operations in IAM identity-based policies, not in bucket policies.

In the policies, the `Resource` element must be `"*"`. For more information about example policies, see Account operations.

The following is the mapping of account operations and required policy actions.

API operations	Policy actions	Description of policy actions
CreateJob	(Required) `s3:CreateJob`	Required to create a new S3 Batch Operations job.
CreateStorageLensGroup	(Required) `s3:CreateStorageLensGroup`	Required to create a new S3 Storage Lens group and associate it with the specified AWS account ID.
	(Conditionally required) `s3:TagResource`	Required if you want to create an S3 Storage Lens group with AWS resource tags.

API operations	Policy actions	Description of policy actions
DeletePublicAccessBlock (Account-level)	(Required) `s3:PutAccountPublicAccessBlock`	Required to remove the block public access configuration from an AWS account.
GetAccessPoint	(Required) `s3:GetAccessPoint`	Required to retrieve configuration information about the specified access point.
GetAccessPointPolicy (Account-level)	(Required) `s3:GetAccountPublicAccessBlock`	Required to retrieve the block public access configuration for an AWS account.
ListAccessPoints	(Required) `s3:ListAccessPoints`	Required to list access points of an S3 bucket that are owned by an AWS account.
ListAccessPointsForObjectLambda	(Required) `s3:ListAccessPointsForObjectLambda`	Required to list the Object Lambda Access Points.
ListBuckets	(Required) `s3:ListAllMyBuckets`	Required to return a list of all buckets that are owned by the authenticated sender of the request.
ListJobs	(Required) `s3:ListJobs`	Required to list current jobs and jobs that have ended recently.
ListMultiRegionAccessPoints	(Required) `s3:ListMultiRegionAccessPoints`	Required to return a list of the Multi-Region Access Points that are currently associated with the specified AWS account.

API operations	Policy actions	Description of policy actions
ListStorageLensConfigurations	(Required) `s3:ListStorageLensConfigurations`	Required to get a list of S3 Storage Lens configurations for an AWS account.
ListStorageLensGroups	(Required) `s3:ListStorageLensGroups`	Required to list all the S3 Storage Lens groups in the specified home AWS Region.
PutPublicAccessBlock (Account-level)	(Required) `s3:PutAccountPublicAccessBlock`	Required to create or modify the block public access configuration for an AWS account.
PutStorageLensConfiguration	(Required) `s3:PutStorageLensConfiguration`	Required to put an S3 Storage Lens configuration.

Policies and permissions in Amazon S3

This page provides an overview of bucket and user policies in Amazon S3 and describes the basic elements of an AWS Identity and Access Management (IAM) policy. Each listed element links to more details about that element and examples of how to use it.

For a complete list of Amazon S3 actions, resources, and conditions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

In its most basic sense, a policy contains the following elements:

- Resource – The Amazon S3 bucket, object, access point, or job that the policy applies to. Use the Amazon Resource Name (ARN) of the bucket, object, access point, or job to identify the resource.

  An example for bucket-level operations:

  `"Resource": "arn:aws:s3:::bucket_name"`

Examples for object-level operations:

- "Resource": "arn:aws:s3:::*bucket_name/**" for all objects in the bucket.
- "Resource": "arn:aws:s3:::*bucket_name/prefix/**" for objects under a certain prefix in the bucket.

For more information, see Policy resources for Amazon S3.

- Actions – For each resource, Amazon S3 supports a set of operations. You identify resource operations that you will allow (or deny) by using action keywords.

  For example, the s3:ListBucket permission allows the user to use the Amazon S3 ListObjectsV2 operation. (The s3:ListBucket permission is a case where the action name doesn't map directly to the operation name.) For more information about using Amazon S3 actions, see Policy actions for Amazon S3. For a complete list of Amazon S3 actions, see Actions in the *Amazon Simple Storage Service API Reference*.

- Effect – What the effect will be when the user requests the specific action—this can be either Allow or Deny.

  If you don't explicitly grant access to (allow) a resource, access is implicitly denied. You can also explicitly deny access to a resource. You might do this to make sure that a user can't access the resource, even if a different policy grants access. For more information, see IAM JSON Policy Elements: Effect in the *IAM User Guide*.

- Principal – The account or user who is allowed access to the actions and resources in the statement. In a bucket policy, the principal is the user, account, service, or other entity that is the recipient of this permission. For more information, see Principals for bucket policies.

- Condition – Conditions for when a policy is in effect. You can use AWS-wide keys and Amazon S3-specific keys to specify conditions in an Amazon S3 access policy. For more information, see Bucket policy examples using condition keys.

The following example bucket policy shows the Effect, Principal, Action, and Resource elements. This policy allows *Akua*, a user in account *123456789012*, s3:GetObject, s3:GetBucketLocation, and s3:ListBucket Amazon S3 permissions on the *amzn-s3-demo-bucket1* bucket.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "ExamplePolicy01",
    "Statement": [
        {
            "Sid": "ExampleStatement01",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Akua"
            },
            "Action": [
                "s3:GetObject",
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        }
    ]
}
```

For complete policy language information, see Policies and permissions in IAM and IAM JSON policy reference in the *IAM User Guide*.

Permission delegation

If an AWS account owns a resource, it can grant those permissions to another AWS account. That account can then delegate those permissions, or a subset of them, to users in the account. This is referred to as *permission delegation*. But an account that receives permissions from another account can't delegate permission cross-account to another AWS account.

Amazon S3 bucket and object ownership

Buckets and objects are Amazon S3 resources. By default, only the resource owner can access these resources. The resource owner refers to the AWS account that creates the resource. For example:

- The AWS account that you use to create buckets and upload objects owns those resources.

- If you upload an object using AWS Identity and Access Management (IAM) user or role credentials, the AWS account that the user or role belongs to owns the object.
- A bucket owner can grant cross-account permissions to another AWS account (or users in another account) to upload objects. In this case, the AWS account that uploads objects owns those objects. The bucket owner doesn't have permissions on the objects that other accounts own, with the following exceptions:
  - The bucket owner pays the bills. The bucket owner can deny access to any objects, or delete any objects in the bucket, regardless of who owns them.
  - The bucket owner can archive any objects or restore archived objects regardless of who owns them. Archival refers to the storage class used to store the objects. For more information, see Managing the lifecycle of objects.

Ownership and request authentication

All requests to a bucket are either authenticated or unauthenticated. Authenticated requests must include a signature value that authenticates the request sender, and unauthenticated requests do not. For more information about request authentication, see Making requests in the *Amazon S3 API Reference*.

A bucket owner can allow unauthenticated requests. For example, unauthenticated PutObject requests are allowed when a bucket has a public bucket policy, or when a bucket ACL grants `WRITE` or `FULL_CONTROL` access to the `All Users` group or the anonymous user specifically. For more information about public bucket policies and public access control lists (ACLs), see The meaning of "public".

All unauthenticated requests are made by the anonymous user. This user is represented in ACLs by the specific canonical user ID `65a011a29cdf8ec533ec3d1ccaae921c`. If an object is uploaded to a bucket through an unauthenticated request, the anonymous user owns the object. The default object ACL grants `FULL_CONTROL` to the anonymous user as the object's owner. Therefore, Amazon S3 allows unauthenticated requests to retrieve the object or modify its ACL.

To prevent objects from being modified by the anonymous user, we recommend that you do not implement bucket policies that allow anonymous public writes to your bucket or use ACLs that allow the anonymous user write access to your bucket. You can enforce this recommended behavior by using Amazon S3 Block Public Access.

For more information about blocking public access, see Blocking public access to your Amazon S3 storage. For more information about ACLs, see Access control list (ACL) overview.

Important

We recommend that you don't use the AWS account root user credentials to make authenticated requests. Instead, create an IAM role and grant that role full access. We refer to users with this role as *administrator users*. You can use credentials assigned to the administrator role, instead of AWS account root user credentials, to interact with AWS and perform tasks, such as create a bucket, create users, and grant permissions. For more information, see AWS security credentials and Security best practices in IAM in the *IAM User Guide*.

Bucket policies for Amazon S3

A bucket policy is a resource-based policy that you can use to grant access permissions to your Amazon S3 bucket and the objects in it. Only the bucket owner can associate a policy with a bucket. The permissions attached to the bucket apply to all of the objects in the bucket that are owned by the bucket owner. These permissions don't apply to objects that are owned by other AWS accounts.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to control ownership of objects uploaded to your bucket and to disable or enable access control lists (ACLs). By default, Object Ownership is set to the Bucket owner enforced setting and all ACLs are disabled. The bucket owner owns all the objects in the bucket and manages access to data exclusively using policies.

Bucket policies use JSON-based AWS Identity and Access Management (IAM) policy language. You can use bucket policies to add or deny permissions for the objects in a bucket. Bucket policies can allow or deny requests based on the elements in the policy. These elements include the requester, S3 actions, resources, and aspects or conditions of the request (such as the IP address that's used to make the request).

For example, you can create a bucket policy that does the following:

- Grants other accounts cross-account permissions to upload objects to your S3 bucket
- Makes sure that you, the bucket owner, has full control of the uploaded objects

For more information, see Examples of Amazon S3 bucket policies.

> **Important**
>
> You can't use a bucket policy to prevent deletions or transitions by an S3 Lifecycle rule. For example, even if your bucket policy denies all actions for all principals, your S3 Lifecycle configuration still functions as normal.

The topics in this section provide examples and show you how to add a bucket policy in the S3 console. For information about identity-based policies, see Identity-based policies for Amazon S3. For information about bucket policy language, see Policies and permissions in Amazon S3.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Topics

- Adding a bucket policy by using the Amazon S3 console
- Controlling access from VPC endpoints with bucket policies
- Examples of Amazon S3 bucket policies
- Bucket policy examples using condition keys

Adding a bucket policy by using the Amazon S3 console

You can use the AWS Policy Generator and the Amazon S3 console to add a new bucket policy or edit an existing bucket policy. A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy. You add a bucket policy to a bucket to grant other AWS accounts or IAM users access permissions for the bucket and the objects in it. Object permissions apply only to the objects that the bucket owner creates. For more information about bucket policies, see Identity and Access Management for Amazon S3.

Make sure to resolve security warnings, errors, general warnings, and suggestions from AWS Identity and Access Management Access Analyzer before you save your policy. IAM Access Analyzer runs policy checks to validate your policy against IAM policy grammar and best practices. These checks generate findings and provide actionable recommendations to help you author policies that are functional and conform to security best practices. To learn more about validating policies by using IAM Access Analyzer, see IAM Access Analyzer policy validation in the *IAM User Guide*. To view a list of the warnings, errors, and suggestions that are returned by IAM Access Analyzer, see IAM Access Analyzer policy check reference.

For guidance on troubleshooting errors with a policy, see Troubleshoot access denied (403 Forbidden) errors in Amazon S3.

To create or edit a bucket policy

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets** or **Directory buckets**.
3. In the list of buckets, choose the name of the bucket that you want to create a bucket policy for or whose bucket policy you want to edit.
4. Choose the **Permissions** tab.
5. Under **Bucket policy**, choose **Edit**. The **Edit bucket policy** page appears.
6. On the **Edit bucket policy** page, do one of the following:

   - To see examples of bucket policies, choose **Policy examples**. Or see Examples of Amazon S3 bucket policies in the *Amazon S3 User Guide*.
   - To generate a policy automatically, or edit the JSON in the **Policy** section, choose **Policy generator**.

   If you choose **Policy generator**, the AWS Policy Generator opens in a new window.

   a. On the **AWS Policy Generator** page, for **Select Type of Policy**, choose **S3 Bucket Policy**.

   b. Add a statement by entering the information in the provided fields, and then choose **Add Statement**. Repeat this step for as many statements as you would like to add. For more information about these fields, see the IAM JSON policy elements reference in the *IAM User Guide*.

   > **Note**
   >
   > For your convenience, the **Edit bucket policy** page displays the **Bucket ARN** (Amazon Resource Name) of the current bucket above the **Policy** text field. You can copy this ARN for use in the statements on the **AWS Policy Generator** page.

   c. After you finish adding statements, choose **Generate Policy**.

   d. Copy the generated policy text, choose **Close**, and return to the **Edit bucket policy** page in the Amazon S3 console.

7. In the **Policy** box, edit the existing policy or paste the bucket policy from the AWS Policy Generator. Make sure to resolve security warnings, errors, general warnings, and suggestions before you save your policy.

   > **Note**
   >
   > Bucket policies are limited to 20 KB in size.

8. (Optional) Choose **Preview external access** in the lower-right corner to preview how your new policy affects public and cross-account access to your resource. Before you save your policy, you can check whether it introduces new IAM Access Analyzer findings or resolves existing findings. If you don't see an active analyzer, choose **Go to Access Analyzer** to create an account analyzer in IAM Access Analyzer. For more information, see Preview access in the *IAM User Guide*.
9. Choose **Save changes**, which returns you to the **Permissions** tab.

Controlling access from VPC endpoints with bucket policies

You can use Amazon S3 bucket policies to control access to buckets from specific virtual private cloud (VPC) endpoints or specific VPCs. This section contains example bucket policies that you can use to control Amazon S3 bucket access from VPC endpoints. To learn how to set up VPC endpoints, see VPC Endpoints in the *VPC User Guide*.

A VPC enables you to launch AWS resources into a virtual network that you define. A VPC endpoint enables you to create a private connection between your VPC and another AWS service. This private connection doesn't require access over the internet, through a virtual private network (VPN) connection, through a NAT instance, or through Direct Connect.

A VPC endpoint for Amazon S3 is a logical entity within a VPC that allows connectivity only to Amazon S3. The VPC endpoint routes requests to Amazon S3 and routes responses back to the VPC. VPC endpoints change only how requests are routed. Amazon S3 public endpoints and DNS names will continue to work with VPC endpoints. For important information about using VPC endpoints with Amazon S3, see Gateway endpoints and Gateway endpoints for Amazon S3 in the *VPC User Guide*.

VPC endpoints for Amazon S3 provide two ways to control access to your Amazon S3 data:

- You can control the requests, users, or groups that are allowed through a specific VPC endpoint. For information about this type of access control, see Controlling access to VPC endpoints using endpoint policies in the *VPC User Guide*.
- You can control which VPCs or VPC endpoints have access to your buckets by using Amazon S3 bucket policies. For examples of this type of bucket policy access control, see the following topics on restricting access.

Topics

- Restricting access to a specific VPC endpoint
- Restricting access to a specific VPC
- Restricting access to an IPv6 VPC endpoint

> **Important**
>
> When applying the Amazon S3 bucket policies for VPC endpoints described in this section, you might block your access to the bucket unintentionally. Bucket permissions that are intended to specifically limit bucket access to connections originating from your VPC endpoint can block all connections to the bucket. For information about how to fix this issue, see How do I fix my bucket policy when it has the wrong VPC or VPC endpoint ID? in the *AWS Support Knowledge Center*.

Restricting access to a specific VPC endpoint

The following is an example of an Amazon S3 bucket policy that restricts access to a specific bucket, `amzn-s3-demo-bucket`, only from the VPC endpoint with the ID `vpce-0abcdef1234567890`. If the specified endpoint is not used, the policy denies all access to the bucket. The `aws:SourceVpce` condition specifies the endpoint. The `aws:SourceVpce` condition doesn't require an Amazon Resource Name (ARN) for the VPC endpoint resource, only the VPC endpoint ID. For more information about using conditions in a policy, see Bucket policy examples using condition keys.

> **Important**
>
> - Before using the following example policy, replace the VPC endpoint ID with an appropriate value for your use case. Otherwise, you won't be able to access your bucket.

- This policy disables console access to the specified bucket because console requests don't originate from the specified VPC endpoint.

JSON

```
{
   "Version":"2012-10-17",
   "Id": "Policy1415115909152",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPCE-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket",
                    "arn:aws:s3:::amzn-s3-demo-bucket/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:SourceVpce": "vpce-0abcdef1234567890"
         }
       }
     }
   ]
}
```

Restricting access to a specific VPC

You can create a bucket policy that restricts access to a specific VPC by using the `aws:SourceVpc` condition. This is useful if you have multiple VPC endpoints configured in the same VPC, and you want to manage access to your Amazon S3 buckets for all of your endpoints. The following is an example of a policy that denies access to `amzn-s3-demo-bucket` and its objects from anyone outside VPC `vpc-1a2b3c4d`. If the specified VPC isn't used, the policy denies all access to the bucket. This statement doesn't grant access to the bucket. To grant access, you must add a separate `Allow` statement. The `vpc-1a2b3c4d` condition key doesn't require an ARN for the VPC resource, only the VPC ID.

Important

- Before using the following example policy, replace the VPC ID with an appropriate value for your use case. Otherwise, you won't be able to access your bucket.
- This policy disables console access to the specified bucket because console requests don't originate from the specified VPC.

JSON

```
{
   "Version":"2012-10-17",
   "Id": "Policy1415115909153",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPC-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket",
                    "arn:aws:s3:::amzn-s3-demo-bucket/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:SourceVpc": "vpc-1a2b3c4d"
         }
       }
     }
   ]
}
```

Restricting access to an IPv6 VPC endpoint

The following example policy denies all Amazon S3 (`s3:`) actions on the *`amzn-s3-demo-bucket`* bucket and its objects, unless the request originates from the specified VPC endpoint (`vpce-0a1b2c3d4e5f6g`) and the source IP address matches the provided IPv6 CIDR block.

```
{
   "Version": "2012-10-17",
   "Id": "Policy1415115909154",
```

```
   "Statement": [
     {
       "Sid": "AccessSpecificIPv6VPCEOnly",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket",
                    "arn:aws:s3:::amzn-s3-demo-bucket/*"],
       "Condition": {
         "StringNotEquals": {
           "aws:SourceVpce": "vpce-0a1b2c3d4e5f6g"
         },
        "NotIpAddress": {
          "aws:VpcSourceIp": "2001:db8::/32"
        }
       }
     }
   ]
}
```

For information on how to restrict access to your bucket based on specific IPs or VPCs, see How do I allow only specific VPC endpoints or IP addresses to access my Amazon S3 bucket? in the AWS re:Post Knowledge Center.

Examples of Amazon S3 bucket policies

With Amazon S3 bucket policies, you can secure access to objects in your buckets, so that only users with the appropriate permissions can access them. You can even prevent authenticated users without the appropriate permissions from accessing your Amazon S3 resources.

This section presents examples of typical use cases for bucket policies. These sample policies use *amzn-s3-demo-bucket* as the resource value. To test these policies, replace the *user input placeholders* with your own information (such as your bucket name).

To grant or deny permissions to a set of objects, you can use wildcard characters (*) in Amazon Resource Names (ARNs) and other values. For example, you can control access to groups of objects that begin with a common prefix or end with a specific extension, such as `.html`.

For more information about AWS Identity and Access Management (IAM) policy language, see Policies and permissions in Amazon S3.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

> **Note**
>
> When testing permissions by using the Amazon S3 console, you must grant additional permissions that the console requires—`s3:ListAllMyBuckets`, `s3:GetBucketLocation`, and `s3:ListBucket`. For an example walkthrough that grants permissions to users and tests those permissions by using the console, see Controlling access to a bucket with user policies.

Additional resources for creating bucket policies include the following:

- For a list of the IAM policy actions, resources, and condition keys that you can use when creating a bucket policy, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.
- For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.
- For guidance on creating your S3 policy, see Adding a bucket policy by using the Amazon S3 console.
- To troubleshoot errors with a policy, see Troubleshoot access denied (403 Forbidden) errors in Amazon S3.

If you're having trouble adding or updating a policy, see Why do I get the error "Invalid principal in policy" when I try to update my Amazon S3 bucket policy? in the AWS re:Post Knowledge Center.

Topics

- Granting read-only permission to a public anonymous user
- Requiring encryption
- Managing buckets using canned ACLs
- Managing object access with object tagging
- Managing object access by using global condition keys
- Managing access based on HTTP or HTTPS requests
- Managing user access to specific folders
- Managing access for access logs
- Managing access to an Amazon CloudFront OAI

- Managing access for Amazon S3 Storage Lens
- Managing permissions for S3 Inventory, S3 analytics, and S3 Inventory reports
- Requiring MFA
- Preventing users from deleting objects

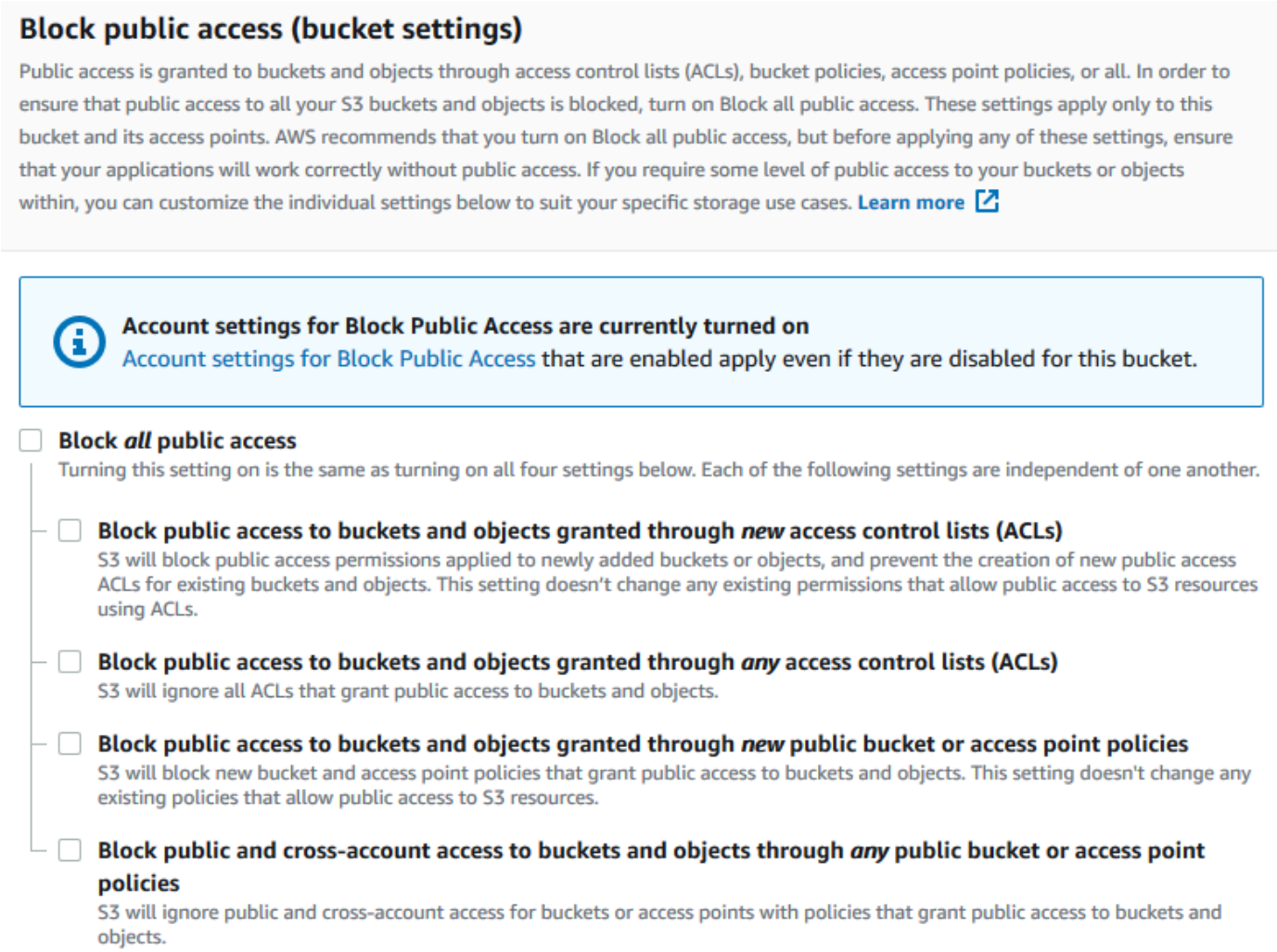
Granting read-only permission to a public anonymous user

You can use your policy settings to grant access to public anonymous users, which is useful if you're configuring your bucket as a static website. Granting access to public anonymous users requires you to disable the Block Public Access settings for your bucket. For more information about how to do this, and the policy required, see Setting permissions for website access. To learn how to set up more restrictive policies for the same purpose, see How can I grant public read access to some objects in my Amazon S3 bucket? in the AWS Knowledge Center.

By default, Amazon S3 blocks public access to your account and buckets. If you want to use a bucket to host a static website, you can use these steps to edit your block public access settings.

> **Warning**
>
> Before you complete these steps, review Blocking public access to your Amazon S3 storage to ensure that you understand and accept the risks involved with allowing public access. When you turn off block public access settings to make your bucket public, anyone on the internet can access your bucket. We recommend that you block all public access to your buckets.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the bucket that you have configured as a static website.
3. Choose **Permissions**.
4. Under **Block public access (bucket settings)**, choose **Edit**.
5. Clear **Block *all* public access**, and choose **Save changes**.



Amazon S3 turns off the Block Public Access settings for your bucket. To create a public static website, you might also have to edit the Block Public Access settings for your account before adding a bucket policy. If the Block Public Access settings for your account are currently turned on, you see a note under **Block public access (bucket settings)**.

Requiring encryption

You can require server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), as shown in the following examples.

Require SSE-KMS for all objects written to a bucket

The following example policy requires every object that is written to the bucket to be encrypted with server-side encryption using AWS Key Management Service (AWS KMS) keys (SSE-KMS). If the object isn't encrypted with SSE-KMS, the request is denied.

JSON

```
{
"Version":"2012-10-17",
"Id": "PutObjPolicy",
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMS",
  "Principal": "*",
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
  "Condition": {
    "Null": {
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "true"
    }
  }
}]
}
```

Require SSE-KMS with a specific AWS KMS key for all objects written to a bucket

The following example policy denies any objects from being written to the bucket if they aren't encrypted with SSE-KMS by using a specific KMS key ID. Even if the objects are encrypted with SSE-KMS by using a per-request header or bucket default encryption, the objects can't be written to the bucket if they haven't been encrypted with the specified KMS key. Make sure to replace the KMS key ARN that's used in this example with your own KMS key ARN.

JSON

```
{
"Version":"2012-10-17",
"Id": "PutObjPolicy",
"Statement": [{
  "Sid": "DenyObjectsThatAreNotSSEKMSWithSpecificKey",
  "Principal": "*",
  "Effect": "Deny",
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
  "Condition": {
    "ArnNotEqualsIfExists": {
```

```
      "s3:x-amz-server-side-encryption-aws-kms-key-id": "arn:aws:kms:us-
east-1:111122223333:key/01234567-89ab-cdef-0123-456789abcdef"
    }
  }
}]
}
```

Managing buckets using canned ACLs

Granting permissions to multiple accounts to upload objects or set object ACLs for public access

The following example policy grants the `s3:PutObject` and `s3:PutObjectAcl` permissions to multiple AWS accounts. Also, the example policy requires that any requests for these operations must include the `public-read` canned access control list (ACL). For more information, see Policy actions for Amazon S3 and Policy condition keys for Amazon S3.

> **Warning**
>
> The `public-read` canned ACL allows anyone in the world to view the objects in your bucket. Use caution when granting anonymous access to your Amazon S3 bucket or disabling block public access settings. When you grant anonymous access, anyone in the world can access your bucket. We recommend that you never grant anonymous access to your Amazon S3 bucket unless you specifically need to, such as with static website hosting. If you want to enable block public access settings for static website hosting, see Tutorial: Configuring a static website on Amazon S3.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AddPublicReadCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root",
                    "arn:aws:iam::444455556666:root"
```

```
                ]
            },
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": [
                        "public-read"
                    ]
                }
            }
        }
    ]
}
```

Grant cross-account permissions to upload objects while ensuring that the bucket owner has full control

The following example shows how to allow another AWS account to upload objects to your bucket while ensuring that you have full control of the uploaded objects. This policy grants a specific AWS account (*111122223333*) the ability to upload objects only if that account includes the `bucket-owner-full-control` canned ACL on upload. The `StringEquals` condition in the policy specifies the `s3:x-amz-acl` condition key to express the canned ACL requirement. For more information, see Policy condition keys for Amazon S3.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
     {
       "Sid":"PolicyForAllowUploadWithACL",
       "Effect":"Allow",
       "Principal":{"AWS":"111122223333"},
       "Action":"s3:PutObject",
       "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*",
       "Condition": {
         "StringEquals": {"s3:x-amz-acl":"bucket-owner-full-control"}
```

```
        }
      }
   ]
}
```

Managing object access with object tagging

Allow a user to read only objects that have a specific tag key and value

The following permissions policy limits a user to only reading objects that have the `environment:production` tag key and value. This policy uses the `s3:ExistingObjectTag` condition key to specify the tag key and value.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Principal":{
            "AWS":"arn:aws:iam::111122223333:role/JohnDoe"
         },
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*",
         "Condition":{
            "StringEquals":{
               "s3:ExistingObjectTag/environment":"production"
            }
         }
      }
   ]
}
```

Restrict which object tag keys that users can add

The following example policy grants a user permission to perform the `s3:PutObjectTagging` action, which allows a user to add tags to an existing object. The condition uses the `s3:RequestObjectTagKeys` condition key to specify the allowed tag keys, such as `Owner` or `CreationDate`. For more information, see Creating a condition that tests multiple key values in the *IAM User Guide*.

The policy ensures that every tag key specified in the request is an authorized tag key. The `ForAnyValue` qualifier in the condition ensures that at least one of the specified keys must be present in the request.

JSON

```
{
   "Version":"2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
            "arn:aws:iam::111122223333:role/JohnDoe"
         ]
       },
 "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {"ForAnyValue:StringEquals": {"s3:RequestObjectTagKeys": [
            "Owner",
            "CreationDate"
          ]
        }
      }
    }
  ]
}
```

Require a specific tag key and value when allowing users to add object tags

The following example policy grants a user permission to perform the `s3:PutObjectTagging` action, which allows a user to add tags to an existing object. The condition requires the user to include a specific tag key (such as *`Project`*) with the value set to *`X`*.

JSON

```
{
   "Version":"2012-10-17",
  "Statement": [
    {"Principal":{"AWS":[
       "arn:aws:iam::111122223333:user/JohnDoe"
         ]
       },
      "Effect": "Allow",
      "Action": [
        "s3:PutObjectTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {"StringEquals": {"s3:RequestObjectTag/Project": "X"
        }
      }
    }
  ]
}
```

Allow a user to only add objects with a specific object tag key and value

The following example policy grants a user permission to perform the `s3:PutObject` action so that they can add objects to a bucket. However, the `Condition` statement restricts the tag keys and values that are allowed on the uploaded objects. In this example, the user can only add objects that have the specific tag key (*`Department`*) with the value set to *`Finance`* to the bucket.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [{
        "Principal":{
            "AWS":[
                 "arn:aws:iam::111122223333:user/JohnDoe"
         ]
        },
        "Effect": "Allow",
        "Action": [
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ],
        "Condition": {
            "StringEquals": {
                "s3:RequestObjectTag/Department": "Finance"
            }
        }
    }]
}
```

Managing object access by using global condition keys

Global condition keys are condition context keys with an `aws` prefix. AWS services can support global condition keys or service-specific keys that include the service prefix. You can use the `Condition` element of a JSON policy to compare the keys in a request with the key values that you specify in your policy.

Restrict access to only Amazon S3 server access log deliveries

In the following example bucket policy, the aws:SourceArn global condition key is used to compare the Amazon Resource Name (ARN) of the resource, making a service-to-service request with the ARN that is specified in the policy. The `aws:SourceArn` global condition key is used to prevent the Amazon S3 service from being used as a confused deputy during transactions between services. Only the Amazon S3 service is allowed to add objects to the Amazon S3 bucket.

This example bucket policy grants `s3:PutObject` permissions to only the logging service principal (`logging.s3.amazonaws.com`).

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowPutObjectS3ServerAccessLogsPolicy",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket-logs/*",
            "Condition": {
                "StringEquals": {
                "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-bucket1"
                }
            }
        },
        {
            "Sid": "RestrictToS3ServerAccessLogs",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket-logs/*",
            "Condition": {
                "ForAllValues:StringNotEquals": {
                    "aws:PrincipalServiceNamesList": "logging.s3.amazonaws.com"
                }
            }
        }
    ]
}
```

Allow access to only your organization

If you want to restrict IAM-based access to a resource so that only principals from AWS accounts in your organization (including the AWS Organizations management account) can access it, you can use the `aws:PrincipalOrgID` global condition key.

Here's an example of a resource-based bucket policy that you can use to grant specific IAM principals in your organization direct access to your bucket. By adding the `aws:PrincipalOrgID` global condition key to your bucket policy, the principal account is now required to be in your organization to obtain access through this policy statement. Even if you accidentally specify an incorrect account when granting access, the `aws:PrincipalOrgID` global condition key acts as an additional safeguard. When this global key is used in an Allow statement, it ensures that only principals from accounts in the listed organization can obtain access through that statement.

> **Important**
>
> This Allow statement with a condition does not deny access granted through other mechanisms. If your bucket has ACLs enabled or other policies that grant access, principals outside your organization may still be able to access objects. To fully restrict access to only your organization, we recommend using this condition in a Deny statement (see example below), disabling ACLs by setting Object Ownership to "Bucket owner enforced", and enabling S3 Block Public Access.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "AllowGetObject",
        "Principal": {
            "AWS": "*"
        },
        "Effect": "Allow",
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "Condition": {
            "StringEquals": {
                "aws:PrincipalOrgID": ["o-aa111bb222"]
            }
```

```
        }
    }]
}
```

Deny access to principals outside your organization

For stronger protection, use a Deny statement with the `aws:PrincipalOrgID` condition. A Deny statement overrides any Allow, regardless of how access is granted. The following example denies all S3 actions to any principal that is not in your organization. This approach provides defense-in-depth even if ACLs or other policies would have inadvertently granted access.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyAccessFromOutsideOrg",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ],
            "Condition": {
                "StringNotEquals": {
                    "aws:PrincipalOrgID": "o-aa111bb222"
                }
            }
        }
    ]
}
```

Managing access based on HTTP or HTTPS requests

Restrict access to only HTTPS requests

If you want to prevent potential attackers from manipulating network traffic, you can use HTTPS (TLS) to only allow encrypted connections while restricting HTTP requests from accessing your bucket. To determine whether the request is HTTP or HTTPS, use the aws:SecureTransport global condition key in your S3 bucket policy. The `aws:SecureTransport` condition key checks whether a request was sent by using HTTP.

If a request returns `true`, then the request was sent through HTTPS. If the request returns `false`, then the request was sent through HTTP. You can then allow or deny access to your bucket based on the desired request scheme.

In the following example, the bucket policy explicitly denies HTTP requests.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "RestrictToTLSRequestsOnly",
        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket",
            "arn:aws:s3:::amzn-s3-demo-bucket/*"
        ],
        "Condition": {
            "Bool": {
                "aws:SecureTransport": "false"
            }
        },
        "Principal": "*"
    }]
}
```

Restrict access to a specific HTTP referer

Suppose that you have a website with the domain name *www.example.com* or *example.com* with links to photos and videos stored in your bucket named *amzn-s3-demo-bucket*. By default, all Amazon S3 resources are private, so only the AWS account that created the resources can access them.

To allow read access to these objects from your website, you can add a bucket policy that allows the `s3:GetObject` permission with a condition that the `GET` request must originate from specific webpages. The following policy restricts requests by using the `StringLike` condition with the `aws:Referer` condition key.

Make sure that the browsers that you use include the HTTP `referer` header in the request.

> **Warning**
>
> We recommend that you use caution when using the `aws:Referer` condition key. It is dangerous to include a publicly known HTTP referer header value. Unauthorized parties can use modified or custom browsers to provide any `aws:Referer` value that they choose. Therefore, do not use `aws:Referer` to prevent unauthorized parties from making direct AWS requests.
>
> The `aws:Referer` condition key is offered only to allow customers to protect their digital content, such as content stored in Amazon S3, from being referenced on unauthorized third-party sites. For more information, see aws:Referer in the *IAM User Guide*.

Managing user access to specific folders

Grant users access to specific folders

Suppose that you're trying to grant users access to a specific folder. If the IAM user and the S3 bucket belong to the same AWS account, then you can use an IAM policy to grant the user access to a specific bucket folder. With this approach, you don't need to update your bucket policy to grant access. You can add the IAM policy to an IAM role that multiple users can switch to.

If the IAM identity and the S3 bucket belong to different AWS accounts, then you must grant cross-account access in both the IAM policy and the bucket policy. For more information about granting cross-account access, see Bucket owner granting cross-account bucket permissions.

The following example bucket policy grants *JohnDoe* full console access to only his folder (home/*JohnDoe*/). By creating a home folder and granting the appropriate permissions to your users, you can have multiple users share a single bucket. This policy consists of three `Allow` statements:

- *AllowRootAndHomeListingOfCompanyBucket*: Allows the user (*JohnDoe*) to list objects at the root level of the *amzn-s3-demo-bucket* bucket and in the home folder. This statement also allows the user to search on the prefix home/ by using the console.
- *AllowListingOfUserFolder*: Allows the user (*JohnDoe*) to list all objects in the home/*JohnDoe*/ folder and any subfolders.
- *AllowAllS3ActionsInUserFolder*: Allows the user to perform all Amazon S3 actions by granting `Read`, `Write`, and `Delete` permissions. Permissions are limited to the bucket owner's home folder.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRootAndHomeListingOfCompanyBucket",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"],
            "Condition": {
                "StringEquals": {
                    "s3:prefix": ["", "home/", "home/JohnDoe"],
                    "s3:delimiter": ["/"]
                }
            }
        },
        {
            "Sid": "AllowListingOfUserFolder",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:ListBucket"],
            "Effect": "Allow",
            "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"],
            "Condition": {
                "StringLike": {
                    "s3:prefix": ["home/JohnDoe/*"]
                }
            }
        },
        {
            "Sid": "AllowAllS3ActionsInUserFolder",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
```

```
                    "arn:aws:iam::111122223333:user/JohnDoe"
                ]
            },
            "Action": ["s3:*"],
            "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/home/JohnDoe/*"]
        }
    ]
}
```

Managing access for access logs

Grant access to Application Load Balancer for enabling access logs

When you enable access logs for Application Load Balancer, you must specify the name of the S3 bucket where the load balancer will store the logs. The bucket must have an attached policy that grants Elastic Load Balancing permission to write to the bucket.

In the following example, the bucket policy grants Elastic Load Balancing (ELB) permission to write the access logs to the bucket:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/
AWSLogs/111122223333/*"
        }
    ]
}
```

> **Note**
>
> Make sure to replace *elb-account-id* with the AWS account ID for Elastic Load Balancing for your AWS Region. For the list of Elastic Load Balancing Regions, see Attach a policy to your Amazon S3 bucket in the *Elastic Load Balancing User Guide*.

If your AWS Region does not appear in the supported Elastic Load Balancing Regions list, use the following policy, which grants permissions to the specified log delivery service.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
       "Principal": {
         "Service": "logdelivery.elasticloadbalancing.amazonaws.com"
          },
      "Effect": "Allow",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/prefix/AWSLogs/111122223333/
*"
    }
  ]
}
```

Then, make sure to configure your Elastic Load Balancing access logs by enabling them. You can verify your bucket permissions by creating a test file.

Managing access to an Amazon CloudFront OAI

Grant permission to an Amazon CloudFront OAI

The following example bucket policy grants a CloudFront origin access identity (OAI) permission to get (read) all objects in your S3 bucket. You can use a CloudFront OAI to allow users to access objects in your bucket through CloudFront but not directly through Amazon S3. For more information, see Restricting access to Amazon S3 content by using an Origin Access Identity in the *Amazon CloudFront Developer Guide*.

The following policy uses the OAI's ID as the policy's `Principal`. For more information about using S3 bucket policies to grant access to a CloudFront OAI, see Migrating from origin access identity (OAI) to origin access control (OAC) in the *Amazon CloudFront Developer Guide*.

To use this example:

- Replace *EH1HDMB1FH2TC* with the OAI's ID. To find the OAI's ID, see the Origin Access Identity page on the CloudFront console, or use ListCloudFrontOriginAccessIdentities in the CloudFront API.
- Replace *amzn-s3-demo-bucket* with the name of your bucket.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity EH1HDMB1FH2TC"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

Managing access for Amazon S3 Storage Lens

Grant permissions for Amazon S3 Storage Lens

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily

metrics report in CSV or Parquet format to a general purpose S3 bucket or export the metrics directly to an AWS-managed S3 table bucket.

S3 Storage Lens can export your aggregated storage usage metrics to an Amazon S3 bucket for further analysis. The bucket where S3 Storage Lens places its metrics exports is known as the *destination bucket*. When setting up your S3 Storage Lens metrics export, you must have a bucket policy for the destination bucket. For more information, see Monitoring your storage activity and usage with Amazon S3 Storage Lens.

The following example bucket policy grants Amazon S3 permission to write objects (PUT requests) to a destination bucket. You use a bucket policy like this on the destination bucket when setting up an S3 Storage Lens metrics export.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "S3StorageLensExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "storage-lens.s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/destination-prefix/
StorageLens/111122223333/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control",
                    "aws:SourceAccount": "111122223333",
                    "aws:SourceArn": "arn:aws:s3:region-
code:111122223333:storage-lens/storage-lens-dashboard-configuration-id"
                }
            }
        }
    ]
}
```

When you're setting up an S3 Storage Lens organization-level metrics export, use the following modification to the previous bucket policy's `Resource` statement.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/destination-prefix/
StorageLens/your-organization-id/*",
```

Managing permissions for S3 Inventory, S3 analytics, and S3 Inventory reports

Grant permissions for S3 Inventory and S3 analytics

S3 Inventory creates lists of the objects in a bucket, and S3 analytics Storage Class Analysis export creates output files of the data used in the analysis. The bucket that the inventory lists the objects for is called the *source bucket*. The bucket where the inventory file or the analytics export file is written to is called a *destination bucket*. When setting up an inventory or an analytics export, you must create a bucket policy for the destination bucket. For more information, see Cataloging and analyzing your data with S3 Inventory and Amazon S3 analytics – Storage Class Analysis.

The following example bucket policy grants Amazon S3 permission to write objects (PUT requests) from the account for the source bucket to the destination bucket. You use a bucket policy like this on the destination bucket when setting up S3 Inventory and S3 analytics export.

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
            "Sid": "InventoryAndAnalyticsExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
            "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ],
            "Condition": {
                "ArnLike": {
```

```
                "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

Control S3 Inventory report configuration creation

Cataloging and analyzing your data with S3 Inventory creates lists of the objects in an S3 bucket and the metadata for each object. The `s3:PutInventoryConfiguration` permission allows a user to create an inventory configuration that includes all object metadata fields that are available by default and to specify the destination bucket to store the inventory. A user with read access to objects in the destination bucket can access all object metadata fields that are available in the inventory report. For more information about the metadata fields that are available in S3 Inventory, see Amazon S3 Inventory list.

To restrict a user from configuring an S3 Inventory report, remove the `s3:PutInventoryConfiguration` permission from the user.

Some object metadata fields in S3 Inventory report configurations are optional, meaning that they're available by default but they can be restricted when you grant a user the `s3:PutInventoryConfiguration` permission. You can control whether users can include these optional metadata fields in their reports by using the `s3:InventoryAccessibleOptionalFields` condition key. For a list of the optional metadata fields available in S3 Inventory, see OptionalFields in the *Amazon Simple Storage Service API Reference*.

> **Note**
>
> For directory buckets, use the `s3express:InventoryAccessibleOptionalFields` condition key instead of `s3:InventoryAccessibleOptionalFields`. For more information about configuring S3 Inventory for directory buckets, see Configuring Amazon S3 Inventory.

To grant a user permission to create an inventory configuration with specific optional metadata fields, use the `s3:InventoryAccessibleOptionalFields` condition key to refine the conditions in your bucket policy.

The following example policy grants a user (*Ana*) permission to create an inventory configuration conditionally. The `ForAllValues:StringEquals` condition in the policy uses the `s3:InventoryAccessibleOptionalFields` condition key to specify the two allowed optional metadata fields, namely `Size` and `StorageClass`. So, when *Ana* is creating an inventory configuration, the only optional metadata fields that she can include are `Size` and `StorageClass`.

JSON

```
{
 "Id": "InventoryConfigPolicy",
 "Version":"2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreationConditionally",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action":
    "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET",
   "Condition": {
    "ForAllValues:StringEquals": {
     "s3:InventoryAccessibleOptionalFields": [
        "Size",
        "StorageClass"
        ]
      }
    }
   }
 ]
}
```

To restrict a user from configuring an S3 Inventory report that includes specific optional metadata fields, add an explicit Deny statement to the bucket policy for the source bucket. The following

example bucket policy denies the user *Ana* from creating an inventory configuration in the source bucket *amzn-s3-demo-source-bucket* that includes the optional `ObjectAccessControlList` or `ObjectOwner` metadata fields. The user *Ana* can still create an inventory configuration with other optional metadata fields.

```
{
 "Id": "InventoryConfigSomeFields",
 "Version": "2012-10-17",
 "Statement": [{
   "Sid": "AllowInventoryCreation",
   "Effect": "Allow",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action": "s3:PutInventoryConfiguration",
   "Resource":
    "arn:aws:s3:::amzn-s3-demo-source-bucket"

  },
  {
   "Sid": "DenyCertainInventoryFieldCreation",
   "Effect": "Deny",
   "Principal": {
    "AWS": "arn:aws:iam::111122223333:user/Ana"
   },
   "Action": "s3:PutInventoryConfiguration",
   "Resource":
     "arn:aws:s3:::amzn-s3-demo-source-bucket",
   "Condition": {
    "ForAnyValue:StringEquals": {
     "s3:InventoryAccessibleOptionalFields": [
        "ObjectOwner",
        "ObjectAccessControlList"
        ]
      }
    }
   }
 ]
}
```

Note

The use of the `s3:InventoryAccessibleOptionalFields` condition key in bucket policies doesn't affect the delivery of inventory reports based on the existing inventory configurations.

Important

We recommend that you use `ForAllValues` with an `Allow` effect or `ForAnyValue` with a `Deny` effect, as shown in the prior examples.
Don't use `ForAllValues` with a `Deny` effect nor `ForAnyValue` with an `Allow` effect, because these combinations can be overly restrictive and block inventory configuration deletion.
To learn more about the `ForAllValues` and `ForAnyValue` condition set operators, see Multivalued context keys in the *IAM User Guide*.

Control S3 Inventory report configuration creation for directory buckets

For directory buckets, you use the `s3express:InventoryAccessibleOptionalFields` condition key in an IAM identity-based policy instead of a bucket policy. The following example IAM policy grants a user permission to create an inventory configuration for a directory bucket. The `ForAllValues:StringEquals` condition uses the `s3express:InventoryAccessibleOptionalFields` condition key to specify the allowed optional metadata fields.

```
{
 "Version": "2012-10-17",
 "Statement": [{
  "Sid": "AllowInventoryCreationForDirectoryBucket",
  "Effect": "Allow",
  "Action": "s3express:PutInventoryConfiguration",
  "Resource": "arn:aws:s3express:region:account-id:bucket/bucket-base-name--zone-id--x-
s3",
  "Condition": {
   "ForAllValues:StringEquals": {
    "s3express:InventoryAccessibleOptionalFields": [
     "Size",
```

```
      "StorageClass"
     ]
    }
   }
  }]
}
```

Requiring MFA

Amazon S3 supports MFA-protected API access, a feature that can enforce multi-factor authentication (MFA) for access to your Amazon S3 resources. Multi-factor authentication provides an extra level of security that you can apply to your AWS environment. MFA is a security feature that requires users to prove physical possession of an MFA device by providing a valid MFA code. For more information, see AWS Multi-Factor Authentication. You can require MFA for any requests to access your Amazon S3 resources.

To enforce the MFA requirement, use the `aws:MultiFactorAuthAge` condition key in a bucket policy. IAM users can access Amazon S3 resources by using temporary credentials issued by the AWS Security Token Service (AWS STS). You provide the MFA code at the time of the AWS STS request.

When Amazon S3 receives a request with multi-factor authentication, the `aws:MultiFactorAuthAge` condition key provides a numeric value that indicates how long ago (in seconds) the temporary credential was created. If the temporary credential provided in the request was not created by using an MFA device, this key value is null (absent). In a bucket policy, you can add a condition to check this value, as shown in the following example.

This example policy denies any Amazon S3 operation on the */taxdocuments* folder in the *amzn-s3-demo-bucket* bucket if the request is not authenticated by using MFA. To learn more about MFA, see Using Multi-Factor Authentication (MFA) in AWS in the *IAM User Guide*.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
```

```
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true }}
      }
    ]
 }
```

The `Null` condition in the `Condition` block evaluates to `true` if the `aws:MultiFactorAuthAge` condition key value is null, indicating that the temporary security credentials in the request were created without an MFA device.

The following bucket policy is an extension of the preceding bucket policy. The following policy includes two policy statements. One statement allows the `s3:GetObject` permission on a bucket (*amzn-s3-demo-bucket*) to everyone. Another statement further restricts access to the *amzn-s3-demo-bucket/taxdocuments* folder in the bucket by requiring MFA.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "DenyInsecureConnections",
        "Effect": "Deny",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
        },
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": { "Null": { "aws:MultiFactorAuthAge": true } }
      },
      {
        "Sid": "AllowGetObject",
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111122223333:root"
        },
        "Action": ["s3:GetObject"],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
```

```
      }
    ]
}
```

You can optionally use a numeric condition to limit the duration for which the `aws:MultiFactorAuthAge` key is valid. The duration that you specify with the `aws:MultiFactorAuthAge` key is independent of the lifetime of the temporary security credential that's used in authenticating the request.

For example, the following bucket policy, in addition to requiring MFA authentication, also checks how long ago the temporary session was created. The policy denies any operation if the `aws:MultiFactorAuthAge` key value indicates that the temporary session was created more than an hour ago (3,600 seconds).

JSON

```
{
    "Version":"2012-10-17",
    "Id": "123",
    "Statement": [
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": {"Null": {"aws:MultiFactorAuthAge": true }}
      },
      {
        "Sid": "",
        "Effect": "Deny",
        "Principal": "*",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/taxdocuments/*",
        "Condition": {"NumericGreaterThan": {"aws:MultiFactorAuthAge": 3600 }}
       },
       {
         "Sid": "",
         "Effect": "Allow",
         "Principal": "*",
         "Action": ["s3:GetObject"],
```

```
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
      }
    ]
}
```

Preventing users from deleting objects

By default, users have no permissions. But as you create policies, you might grant users permissions that you didn't intend to grant. To avoid such permission loopholes, you can write a stricter access policy by adding an explicit deny.

To explicitly block users or accounts from deleting objects, you must add the following actions to a bucket policy: `s3:DeleteObject`, `s3:DeleteObjectVersion`, and `s3:PutLifecycleConfiguration` permissions. All three actions are required because you can delete objects either by explicitly calling the `DELETE Object` API operations or by configuring their lifecycle (see Managing the lifecycle of objects) so that Amazon S3 can remove the objects when their lifetime expires.

In the following policy example, you explicitly deny `DELETE Object` permissions to the user *`MaryMajor`*. An explicit Deny statement always supersedes any other permission granted.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "statement1",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/MaryMajor"
      },
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetBucketAcl"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1",
   "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
```

```
    },
    {
      "Sid": "statement2",
      "Effect": "Deny",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/MaryMajor"
      },
      "Action": [
        "s3:DeleteObject",
        "s3:DeleteObjectVersion",
        "s3:PutLifecycleConfiguration"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1",
     "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    }
  ]
}
```

Bucket policy examples using condition keys

You can use access policy language to specify conditions when you grant permissions. You can use the optional `Condition` element, or `Condition` block, to specify conditions for when a policy is in effect.

For policies that use Amazon S3 condition keys for object and bucket operations, see the following examples. For more information about condition keys, see Policy condition keys for Amazon S3. For a complete list of Amazon S3 actions, condition keys, and resources that you can specify in policies, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Examples: Amazon S3 condition keys for object operations

The following examples show how you can use Amazon S3-specific condition keys for object operations. For a complete list of Amazon S3 actions, condition keys, and resources that you can specify in policies, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Several of the example policies show how you can use conditions keys with PUT Object operations. PUT Object operations allow access control list (ACL)–specific headers that you can use to grant ACL-based permissions. By using these condition keys, you can set a condition to require specific access permissions when the user uploads an object. You can also grant ACL–based permissions with the PutObjectAcl operation. For more information, see PutObjectAcl in the *Amazon S3 Amazon Simple Storage Service API Reference*. For more information about ACLs, see Access control list (ACL) overview.

Topics

- Example 1: Granting s3:PutObject permission requiring that objects be stored using server-side encryption
- Example 2: Granting s3:PutObject permission to copy objects with a restriction on the copy source
- Example 3: Granting access to a specific version of an object
- Example 4: Granting permissions based on object tags
- Example 5: Restricting access by the AWS account ID of the bucket owner
- Example 6: Requiring a minimum TLS version
- Example 7: Excluding certain principals from a Deny statement
- Example 8: Enforcing clients to conditionally upload objects based on object key names or ETags
- Example 9: Restricting annotation listing by name prefix
- Example 10: Requiring conditional writes for annotation operations
- Example 11: Limiting pagination size for ListObjectAnnotations

Example 1: Granting `s3:PutObject` permission requiring that objects be stored using server-side encryption

Suppose that Account A owns a bucket. The account administrator wants to grant Jane, a user in Account A, permission to upload objects with the condition that Jane always request server-side encryption with Amazon S3 managed keys (SSE-S3). The Account A administrator can specify this requirement by using the `s3:x-amz-server-side-encryption` condition key as shown. The key-value pair in the following `Condition` block specifies the `s3:x-amz-server-side-encryption` condition key and SSE-S3 (`AES256`) as the encryption type:

```
"Condition": {
     "StringNotEquals": {
         "s3:x-amz-server-side-encryption": "AES256"
     }}
```

When testing this permission by using the AWS CLI, you must add the required encryption by using the `--server-side-encryption` parameter, as shown in the following example. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key HappyFace.jpg --body c:
\HappyFace.jpg --server-side-encryption "AES256" --profile AccountAadmin
```

Example 2: Granting `s3:PutObject` permission to copy objects with a restriction on the copy source

In a PUT object request, when you specify a source object, the request is a copy operation (see CopyObject). Accordingly, the bucket owner can grant a user permission to copy objects with restrictions on the source, for example:

- Allow copying objects only from the specified source bucket (for example, *amzn-s3-demo-source-bucket*).
- Allow copying objects from the specified source bucket and only the objects whose key name prefix starts with as specific prefix, such as *public/* (for example, *amzn-s3-demo-source-bucket/public/**).
- Allow copying only a specific object from the source bucket (for example, *amzn-s3-demo-source-bucket/example.jpg*).

The following bucket policy grants a user (*Dave*) the `s3:PutObject` permission. This policy allows him to copy objects only with a condition that the request include the `s3:x-amz-copy-source` header and that the header value specify the */amzn-s3-demo-source-bucket/public/** key name prefix. To use this example policy, replace the *`user input placeholders`* with your own information.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
       {
            "Sid": "cross-account permission to user in your own account",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
        },
        {
            "Sid": "Deny your user permission to upload object if copy source is
 not /bucket/prefix",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-source-bucket/*",
            "Condition": {
                "StringNotLike": {
                    "s3:x-amz-copy-source": "amzn-s3-demo-source-bucket/public/*"
                }
            }
        }
    ]
}
```

Test the policy with the AWS CLI

You can test the permission using the AWS CLI `copy-object` command. You specify the source by adding the `--copy-source` parameter; the key name prefix must match the prefix allowed in the policy. You need to provide the user Dave credentials using the `--profile` parameter. For more information about setting up the AWS CLI, see Developing with Amazon S3 using the AWS CLI in the *Amazon S3 API Reference*.

```
aws s3api copy-object --bucket amzn-s3-demo-source-bucket --key HappyFace.jpg
--copy-source amzn-s3-demo-source-bucket/public/PublicHappyFace1.jpg --
profile AccountADave
```

Give permission to copy only a specific object

The preceding policy uses the `StringNotLike` condition. To grant permission to copy only a specific object, you must change the condition from `StringNotLike` to `StringNotEquals` and then specify the exact object key, as shown in the following example. To use this example command, replace the *`user input placeholders`* with your own information.

```
"Condition": {
       "StringNotEquals": {
           "s3:x-amz-copy-source": "amzn-s3-demo-source-bucket/public/
PublicHappyFace1.jpg"
       }
}
```

Example 3: Granting access to a specific version of an object

Suppose that Account A owns a versioning-enabled bucket. The bucket has several versions of the *`HappyFace.jpg`* object. The Account A administrator now wants to grant the user *`Dave`* permission to get only a specific version of the object. The account administrator can accomplish this by granting the user *`Dave`* the `s3:GetObjectVersion` permission conditionally, as shown in the following example. The key-value pair in the `Condition` block specifies the `s3:VersionId` condition key. In this case, to retrieve the object from the specified versioning-enabled bucket, *`Dave`* needs to know the exact object version ID. To use this example policy, replace the *`user input placeholders`* with your own information.

For more information, see GetObject in the *Amazon Simple Storage Service API Reference*.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/HappyFace.jpg"
        },
        {
```

```
            "Sid": "statement2",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/Dave"
            },
            "Action": "s3:GetObjectVersion",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/HappyFace.jpg",
            "Condition": {
                "StringNotEquals": {
                    "s3:VersionId": "AaaHbAQitwiL_h47_44lRO2DDfLlBO5e"
                }
            }
        }
    ]
}
```

Test the policy with the AWS CLI

You can test the permissions in this policy by using the AWS CLI `get-object` command with the `--version-id` parameter to identify the specific object version to retrieve. The command retrieves the specified version of the object and saves it to the *`OutputFile.jpg`* file.

```
aws s3api get-object --bucket amzn-s3-demo-bucket --key HappyFace.jpg OutputFile.jpg --
version-id AaaHbAQitwiL_h47_44lRO2DDfLlBO5e --profile AccountADave
```

Example 4: Granting permissions based on object tags

For examples of how to use object tagging condition keys with Amazon S3 operations, see Tagging and access control policies.

Example 5: Restricting access by the AWS account ID of the bucket owner

You can use either the `aws:ResourceAccount` or `s3:ResourceAccount` condition key to write IAM or virtual private cloud (VPC) endpoint policies that restrict user, role, or application access to the Amazon S3 buckets that are owned by a specific AWS account ID. You can use these condition keys to restrict clients within your VPC from accessing buckets that you don't own.

However, be aware that some AWS services rely on access to AWS managed buckets. Therefore, using the `aws:ResourceAccount` or `s3:ResourceAccount` key in your IAM policy might also affect access to these resources. For more information, see the following resources:

- Restrict access to buckets in a specified AWS account in the *AWS PrivateLink Guide*
- Restrict access to buckets that Amazon ECR uses in the *Amazon ECR Guide*
- Provide required access to Systems Manager for AWS managed Amazon S3 buckets in the *AWS Systems Manager Guide*

For more information about the `aws:ResourceAccount` and `s3:ResourceAccount` condition keys and examples that show how to use them, see Limit access to Amazon S3 buckets owned by specific AWS accounts in the *AWS Storage Blog*.

Example 6: Requiring a minimum TLS version

You can use the `s3:TlsVersion` condition key to write IAM, virtual private cloud endpoint (VPCE), or bucket policies that restrict user or application access to Amazon S3 buckets based on the TLS version that's used by the client. You can use this condition key to write policies that require a minimum TLS version.

Note

When AWS services make calls to other AWS services on your behalf (service-to-service calls), certain network-specific authorization context is redacted, including `s3:TlsVersion`, `aws:SecureTransport`, `aws:SourceIp`, and `aws:VpcSourceIp`. If your policy uses these condition keys with `Deny` statements, AWS service principals might be unintentionally blocked. To allow AWS services to work properly while maintaining your security requirements, exclude service principals from your `Deny` statements by adding the `aws:PrincipalIsAWSService` condition key with a value of `false`. For example:

```
{
  "Effect": "Deny",
  "Action": "s3:*",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "aws:SecureTransport": "false",
      "aws:PrincipalIsAWSService": "false"
    }
  }
}
```

This policy denies access to S3 operations when HTTPS is not used (`aws:SecureTransport` is false), but only for non-AWS service principals. This ensures your conditional restrictions apply to all principals except AWS service principals.

Example

The following example bucket policy *denies* `PutObject` requests by clients that have a TLS version earlier than 1.2, for example, 1.1 or 1.0. To use this example policy, replace the *`user input placeholders`* with your own information.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ],
            "Condition": {
                "NumericLessThan": {
                    "s3:TlsVersion": 1.2
                }
            }
        }
    ]
}
```

Example

The following example bucket policy *allows* `PutObject` requests by clients that have a TLS version later than 1.1, for example, 1.2, 1.3, or later:

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ],
            "Condition": {
                "NumericGreaterThan": {
                    "s3:TlsVersion": 1.1
                }
            }
        }
    ]
}
```

Example 7: Excluding certain principals from a Deny statement

The following bucket policy denies `s3:GetObject` access to the *amzn-s3-demo-bucket*, except to principals with the account number *123456789012*. To use this example policy, replace the *user input placeholders* with your own information.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessFromPrincipalNotInSpecificAccount",
      "Principal": {
        "AWS": "*"
      },
```

```
      "Action": "s3:GetObject",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalAccount": [
            "123456789012"
          ]
        }
      }
    }
  ]
}
```

Example 8: Enforcing clients to conditionally upload objects based on object key names or ETags

With conditional writes, you can add an additional header to your `WRITE` requests in order to specify preconditions for your S3 operation. This header specifies a condition that, if not met, will result in the S3 operation failing. For example you can prevent overwrites of existing data by validating there is no object with the same key name already in your bucket during object upload. You can alternatively check an object's entity tag (ETag) in Amazon S3 before writing an object.

For bucket policy examples that use conditions in a bucket policy to enforce conditional writes, see the section called “Enforce conditional writes”.

Example 9: Restricting annotation listing by name prefix

The following bucket policy grants the `s3user` user permission to list only the annotations whose names start with `ml-output` on objects in `amzn-s3-demo-bucket`. The `s3:annotation-prefix` condition key applies only to the `ListObjectAnnotations` and `ListObjectVersionAnnotations` actions.

> **Note**
>
> You can't use `s3:annotation-prefix` with `PutObjectAnnotation`, `GetObjectAnnotation`, or `DeleteObjectAnnotation`. If a policy statement combines

this condition key with those actions, IAM rejects the policy with the error `Conditions do not apply to combination of actions and resources in statement`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/s3user"
            },
            "Action": [
                "s3:ListObjectAnnotations",
                "s3:ListObjectVersionAnnotations"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringLike": {
                    "s3:annotation-prefix": "ml-output*"
                }
            }
        }
    ]
}
```

Example 10: Requiring conditional writes for annotation operations

The following bucket policy denies `PutObjectAnnotation` and `DeleteObjectAnnotation` requests unless the caller includes the `x-amz-object-if-match` conditional header. This enforces optimistic concurrency control for all annotation writes on the bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireConditionalWritesForAnnotations",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:PutObjectAnnotation",
                "s3:DeleteObjectAnnotation"
```

```
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "Null": {
                    "s3:x-amz-object-if-match": "true"
                }
            }
        }
    ]
}
```

Example 11: Limiting pagination size for ListObjectAnnotations

The following bucket policy limits the `max-annotation-results` parameter to 100 or fewer for all `ListObjectAnnotations` requests. This prevents callers from requesting excessively large result sets.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "LimitAnnotationListPageSize",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:ListObjectAnnotations",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "NumericGreaterThan": {
                    "s3:max-annotation-results": "100"
                }
            }
        }
    ]
}
```

Examples: Amazon S3 condition keys for bucket operations

The following example policies show how you can use Amazon S3 specific condition keys for bucket operations.

Topics

- Example 1: Granting s3:GetObject permission with a condition on an IP address

- Example 2: Getting a list of objects in a bucket with a specific prefix
- Example 3: Setting the maximum number of keys

Example 1: Granting `s3:GetObject` permission with a condition on an IP address

You can give authenticated users permission to use the `s3:GetObject` action if the request originates from a specific range of IP addresses (for example, *`192.0.2.*`*), unless the IP address is one that you want to exclude (for example, *`192.0.2.188`*). In the `Condition` block, `IpAddress` and `NotIpAddress` are conditions, and each condition is provided a key-value pair for evaluation. Both of the key-value pairs in this example use the `aws:SourceIp` AWS wide key. To use this example policy, replace the *`user input placeholders`* with your own information.

> **Note**
>
> The `IPAddress` and `NotIpAddress` key values specified in the `Condition` block use CIDR notation, as described in RFC 4632. For more information, see http://www.rfc-editor.org/rfc/rfc4632.txt.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "S3PolicyId1",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": "*",
            "Action":"s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition" : {
                "IpAddress" : {
                    "aws:SourceIp": "192.0.2.0/24"
                },
                "NotIpAddress" : {
                    "aws:SourceIp": "192.0.2.188/32"
                }
            }
```

```
        }
    ]
}
```

You can also use other AWS-wide condition keys in Amazon S3 policies. For example, you can specify the `aws:SourceVpce` and `aws:SourceVpc` condition keys in bucket policies for VPC endpoints. For specific examples, see Controlling access from VPC endpoints with bucket policies.

> **Note**
>
> For some AWS global condition keys, only certain resource types are supported. Therefore, check whether Amazon S3 supports the global condition key and resource type that you want to use, or if you'll need to use an Amazon S3 specific condition key instead. For a complete list of supported resource types and condition keys for Amazon S3, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.
> For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Example 2: Getting a list of objects in a bucket with a specific prefix

You can use the `s3:prefix` condition key to limit the response of the ListObjectsV2 API operation to key names with a specific prefix. If you are the bucket owner, you can use this condition key to restrict a user to list the contents of a specific prefix in the bucket. The `s3:prefix` condition key is useful if the objects in the bucket are organized by key name prefixes.

The Amazon S3 console uses key name prefixes to show a folder concept. Only the console supports the concept of folders; the Amazon S3 API supports only buckets and objects. For example, if you have two objects with the key names *`public/object1.jpg`* and *`public/object2.jpg`*, the console shows the objects under the *`public`* folder. In the Amazon S3 API, these are objects with prefixes, not objects in folders. For more information about using prefixes and delimiters to filter access permissions, see Controlling access to a bucket with user policies.

In the following scenario, the bucket owner and the parent account to which the user belongs are the same. So the bucket owner can use either a bucket policy or a user policy to grant access. For more information about other condition keys that you can use with the `ListObjectsV2` API operation, see ListObjectsV2.

Note

If the bucket is versioning-enabled, to list the objects in the bucket, you must grant the `s3:ListBucketVersions` permission in the following policies, instead of the `s3:ListBucket` permission. The `s3:ListBucketVersions` permission also supports the `s3:prefix` condition key.

User policy

The following user policy grants the `s3:ListBucket` permission (see ListObjectsV2) with a `Condition` statement that requires the user to specify a prefix in the request with a value of *`projects`*. To use this example policy, replace the *`user input placeholders`* with your own information.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:ListBucket",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Action": "s3:ListBucket",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
         "Condition" : {
             "StringNotEquals" : {
                 "s3:prefix": "projects"
             }
          }
```

```
        }
    ]
}
```

The `Condition` statement restricts the user to listing only object keys that have the *`projects`* prefix. The added explicit `Deny` statement denies the user from listing keys with any other prefix, no matter what other permissions the user might have. For example, it's possible that the user could get permission to list object keys without any restriction, either through updates to the preceding user policy or through a bucket policy. Because explicit `Deny` statements always override `Allow` statements, if the user tries to list keys other than those that have the *`projects`* prefix, the request is denied.

Bucket policy

If you add the `Principal` element to the above user policy, identifying the user, you now have a bucket policy, as shown in the following example. To use this example policy, replace the *`user input placeholders`* with your own information.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
         },
         "Action":  "s3:ListBucket",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
         "Condition" : {
             "StringEquals" : {
                 "s3:prefix": "projects"
             }
          }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
```

```
            "Principal": {
               "AWS": "arn:aws:iam::123456789012:user/bucket-owner"
            },
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition" : {
                "StringNotEquals" : {
                    "s3:prefix": "projects"
                }
             }
          }
       ]
}
```

Test the policy with the AWS CLI

You can test the policy using the following `list-object` AWS CLI command. In the command, you provide user credentials using the `--profile` parameter. For more information about setting up and using the AWS CLI, see Developing with Amazon S3 using the AWS CLI in the *Amazon S3 API Reference*.

```
aws s3api list-objects --bucket amzn-s3-demo-bucket --prefix projects --
profile AccountA
```

Example 3: Setting the maximum number of keys

You can use the `s3:max-keys` condition key to set the maximum number of keys that a requester can return in a ListObjectsV2 or ListObjectVersions request. By default, these API operations return up to 1,000 keys. For a list of numeric condition operators that you can use with `s3:max-keys` and accompanying examples, see Numeric Condition Operators in the *IAM User Guide*.

Identity-based policies for Amazon S3

By default, users and roles don't have permission to create or modify Amazon S3 resources. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Create IAM policies (console) in the *IAM User Guide*.

For details about actions and resource types defined by Amazon S3, including the format of the ARNs for each of the resource types, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Topics

- Policy best practices
- Controlling access to a bucket with user policies
- Identity-based policy examples for Amazon S3

Policy best practices

Identity-based policies determine whether someone can create, access, or delete Amazon S3 resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.
- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies

adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see Validate policies with IAM Access Analyzer in the *IAM User Guide*.

- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Secure API access with MFA in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Controlling access to a bucket with user policies

This walkthrough explains how user permissions work with Amazon S3. In this example, you create a bucket with folders. You then create AWS Identity and Access Management IAM users in your AWS account and grant those users incremental permissions on your Amazon S3 bucket and the folders in it.

Topics

- Basics of buckets and folders
- Walkthrough summary
- Preparing for the walkthrough
- Step 1: Create a bucket
- Step 2: Create IAM users and a group
- Step 3: Verify that IAM users have no permissions
- Step 4: Grant group-level permissions
- Step 5: Grant IAM user Alice specific permissions
- Step 6: Grant IAM user Bob specific permissions
- Step 7: Secure the private folder
- Step 8: Clean up
- Related resources

Basics of buckets and folders

The Amazon S3 data model is a flat structure: You create a bucket, and the bucket stores objects. There is no hierarchy of subbuckets or subfolders, but you can emulate a folder hierarchy. Tools like the Amazon S3 console can present a view of these logical folders and subfolders in your bucket.

The console shows that a bucket named `companybucket` has three folders, `Private`, `Development`, and `Finance`, and an object, `s3-dg.pdf`. The console uses the object names (keys) to create a logical hierarchy with folders and subfolders. Consider the following examples:

- When you create the `Development` folder, the console creates an object with the key `Development/`. Note the trailing slash (/) delimiter.
- When you upload an object named `Projects1.xls` in the `Development` folder, the console uploads the object and gives it the key `Development/Projects1.xls`.

  In the key, `Development` is the prefix and / is the delimiter. The Amazon S3 API supports prefixes and delimiters in its operations. For example, you can get a list of all objects from a bucket with a specific prefix and delimiter. On the console, when you open the `Development` folder, the console lists the objects in that folder. In the following example, the `Development` folder contains one object.

  When the console lists the `Development` folder in the `companybucket` bucket, it sends a request to Amazon S3 in which it specifies a prefix of `Development` and a delimiter of / in the request. The console's response looks just like a folder list in your computer's file system. The preceding example shows that the bucket `companybucket` has an object with the key `Development/Projects1.xls`.

The console is using object keys to infer a logical hierarchy. Amazon S3 has no physical hierarchy. Amazon S3 only has buckets that contain objects in a flat file structure. When you create objects using the Amazon S3 API, you can use object keys that imply a logical hierarchy. When you create a logical hierarchy of objects, you can manage access to individual folders, as this walkthrough demonstrates.

Before you start, be sure that you are familiar with the concept of the *root-level* bucket content. Suppose that your `companybucket` bucket has the following objects:

- `Private/privDoc1.txt`
- `Private/privDoc2.zip`

- Development/project1.xls
- Development/project2.xls
- Finance/Tax2011/document1.pdf
- Finance/Tax2011/document2.pdf
- s3-dg.pdf

These object keys create a logical hierarchy with Private, Development, and the Finance as root-level folders and s3-dg.pdf as a root-level object. When you choose the bucket name on the Amazon S3 console, the root-level items appear. The console shows the top-level prefixes (Private/, Development/, and Finance/) as root-level folders. The object key s3-dg.pdf has no prefix, and so it appears as a root-level item.

Walkthrough summary

In this walkthrough, you create a bucket with three folders (Private, Development, and Finance) in it.

You have two users, Alice and Bob. You want Alice to access only the Development folder, and you want Bob to access only the Finance folder. You want to keep the Private folder content private. In the walkthrough, you manage access by creating IAM users (the example uses the usernames Alice and Bob) and granting them the necessary permissions.

IAM also supports creating user groups and granting group-level permissions that apply to all users in the group. This helps you better manage permissions. For this exercise, both Alice and Bob need some common permissions. So you also create a group named Consultants and then add both Alice and Bob to the group. You first grant permissions by attaching a group policy to the group. Then you add user-specific permissions by attaching policies to specific users.

> **Note**
>
> The walkthrough uses companybucket as the bucket name, Alice and Bob as the IAM users, and Consultants as the group name. Because Amazon S3 requires that bucket names be globally unique, you must replace the bucket name with a name that you create.

Preparing for the walkthrough

In this example, you use your AWS account credentials to create IAM users. Initially, these users have no permissions. You incrementally grant these users permissions to perform specific Amazon S3 actions. To test these permissions, you sign in to the console with each user's credentials. As you incrementally grant permissions as an AWS account owner and test permissions as an IAM user, you need to sign in and out, each time using different credentials. You can do this testing with one browser, but the process will go faster if you can use two different browsers. Use one browser to connect to the AWS Management Console with your AWS account credentials and another browser to connect with the IAM user credentials.

To sign in to the AWS Management Console with your AWS account credentials, go to https://console.aws.amazon.com/. An IAM user can't sign in using the same link. An IAM user must use an IAM-enabled sign-in page. As the account owner, you can provide this link to your users.

For more information about IAM, see The AWS Management Console Sign-in Page in the *IAM User Guide*.

To provide a sign-in link for IAM users

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the **Navigation** pane, choose **IAM Dashboard** .
3. Note the URL under **IAM users sign in link:**. You will give this link to IAM users to sign in to the console with their IAM user name and password.

Step 1: Create a bucket

In this step, you sign in to the Amazon S3 console with your AWS account credentials, create a bucket, add folders to the bucket, and upload one or two sample documents in each folder.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create a bucket.

   For step-by-step instructions, see Creating a general purpose bucket.
3. Upload one document to the bucket.

This exercise assumes that you have the `s3-dg.pdf` document at the root level of this bucket. If you upload a different document, substitute its file name for `s3-dg.pdf`.

4. Add three folders named `Private`, `Finance`, and `Development` to the bucket.

   For step-by-step instructions to create a folder, see Organizing objects in the Amazon S3 console by using folders> in the *Amazon Simple Storage Service User Guide*.

5. Upload one or two documents to each folder.

   For this exercise, assume that you have uploaded a couple of documents in each folder, resulting in the bucket having objects with the following keys:

   - `Private/privDoc1.txt`
   - `Private/privDoc2.zip`
   - `Development/project1.xls`
   - `Development/project2.xls`
   - `Finance/Tax2011/document1.pdf`
   - `Finance/Tax2011/document2.pdf`
   - `s3-dg.pdf`

   For step-by-step instructions, see Uploading objects.

Step 2: Create IAM users and a group

Now use the IAM Console to add two IAM users, Alice and Bob, to your AWS account. For step-by-step instructions, see Creating an IAM user in your AWS account in the *IAM User Guide*.

Also create an administrative group named `Consultants`. Then add both users to the group. For step-by-step instructions, see Creating IAM user groups.

> **Warning**
>
> When you add users and a group, do not attach any policies that grant permissions to these users. At first, these users don't have any permissions. In the following sections, you grant permissions incrementally. First you must ensure that you have assigned passwords to

these IAM users. You use these user credentials to test Amazon S3 actions and verify that the permissions work as expected.

For step-by-step instructions for creating a new IAM user, see Creating an IAM user in your AWS account in the *IAM User Guide*. When you create the users for this walkthrough, select **AWS Management Console access** and clear programmatic access.

For step-by-step instructions for creating an administrative group, see Creating Your First IAM Admin User and Group in the *IAM User Guide*.

Step 3: Verify that IAM users have no permissions

If you are using two browsers, you can now use the second browser to sign in to the console using one of the IAM user credentials.

1. Using the IAM user sign-in link (see To provide a sign-in link for IAM users), sign in to the AWS Management Console using either of the IAM user credentials.
2. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   Verify the console message telling you that access is denied.

Now, you can begin granting incremental permissions to the users. First, you attach a group policy that grants permissions that both users must have.

Step 4: Grant group-level permissions

You want the users to be able to do the following:

- List all buckets owned by the parent account. To do so, Bob and Alice must have permission for the `s3:ListAllMyBuckets` action.
- List root-level items, folders, and objects in the `companybucket` bucket. To do so, Bob and Alice must have permission for the `s3:ListBucket` action on the `companybucket` bucket.

First, you create a policy that grants these permissions, and then you attach it to the `Consultants` group.

Step 4.1: Grant permission to list all buckets

In this step, you create a managed policy that grants the users minimum permissions to enable them to list all buckets owned by the parent account. Then you attach the policy to the `Consultants` group. When you attach the managed policy to a user or a group, you grant the user or group permission to obtain a list of buckets owned by the parent AWS account.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

   > **Note**
   >
   > Because you are granting user permissions, sign in using your AWS account credentials, not as an IAM user.

2. Create the managed policy.

   a. In the navigation pane on the left, choose **Policies**, and then choose **Create Policy**.

   b. Choose the **JSON** tab.

   c. Copy the following access policy and paste it into the policy text field.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGroupToSeeBucketListInTheConsole",
      "Action": ["s3:ListAllMyBuckets"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]
    }
  ]
}
```

   A policy is a JSON document. In the document, a `Statement` is an array of objects, each describing a permission using a collection of name-value pairs. The preceding policy describes one specific permission. The `Action` specifies the type of access. In the policy,

the `s3:ListAllMyBuckets` is a predefined Amazon S3 action. This action covers the Amazon S3 GET Service operation, which returns a list of all buckets owned by the authenticated sender. The `Effect` element value determines whether specific permission is allowed or denied.

d. Choose **Review Policy**. On the next page, enter `AllowGroupToSeeBucketListInTheConsole` in the **Name** field, and then choose **Create policy**.

> **Note**
>
> The **Summary** entry displays a message stating that the policy does not grant any permissions. For this walkthrough, you can safely ignore this message.

3. Attach the `AllowGroupToSeeBucketListInTheConsole` managed policy that you created to the `Consultants` group.

   For step-by-step instructions for attaching a managed policy, see Adding and removing IAM identity permissions in the *IAM User Guide*.

   You attach policy documents to IAM users and groups in the IAM console. Because you want both users to be able to list the buckets, you attach the policy to the group.

4. Test the permission.

   a. Using the IAM user sign-in link (see To provide a sign-in link for IAM users), sign in to the console using any one of IAM user credentials.

   b. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

      The console should now list all the buckets but not the objects in any of the buckets.

Step 4.2: Enable users to list root-level content of a bucket

Next, you allow all users in the `Consultants` group to list the root-level `companybucket` bucket items. When a user chooses the company bucket on the Amazon S3 console, the user can see the root-level items in the bucket.

> **Note**
>
> This example uses `companybucket` for illustration. You must use the name of the bucket that you created.

To understand the request that the console sends to Amazon S3 when you choose a bucket name, the response that Amazon S3 returns, and how the console interprets the response, examine the flow a little more closely.

When you choose a bucket name, the console sends the GET Bucket (List Objects) request to Amazon S3. This request includes the following parameters:

- The `prefix` parameter with an empty string as its value.
- The `delimiter` parameter with `/` as its value.

The following is an example request.

```
GET ?prefix=&delimiter=/ HTTP/1.1
Host: companybucket.s3.amazonaws.com
Date: Wed, 01 Aug  2012 12:00:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
```

Amazon S3 returns a response that includes the following `<ListBucketResult/>` element.

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix></Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>s3-dg.pdf</Key>
    ...
  </Contents>
  <CommonPrefixes>
    <Prefix>Development/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>Finance/</Prefix>
  </CommonPrefixes>
```

```
  <CommonPrefixes>
    <Prefix>Private/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

The key `s3-dg.pdf` object does not contain the slash (/) delimiter, and Amazon S3 returns the key in the `<Contents>` element. However, all other keys in the example bucket contain the / delimiter. Amazon S3 groups these keys and returns a `<CommonPrefixes>` element for each of the distinct prefix values `Development/`, `Finance/`, and `Private/` that is a substring from the beginning of these keys to the first occurrence of the specified / delimiter.

The console interprets this result and displays the root-level items as three folders and one object key.

If Bob or Alice opens the **Development** folder, the console sends the GET Bucket (List Objects) request to Amazon S3 with the `prefix` and the `delimiter` parameters set to the following values:

- The `prefix` parameter with the value `Development/`.
- The `delimiter` parameter with the "/" value.

In response, Amazon S3 returns the object keys that start with the specified prefix.

```
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>companybucket</Name>
  <Prefix>Development</Prefix>
  <Delimiter>/</Delimiter>
   ...
  <Contents>
    <Key>Project1.xls</Key>
    ...
  </Contents>
  <Contents>
    <Key>Project2.xls</Key>
    ...
  </Contents>
</ListBucketResult>
```

The console shows the object keys.

Now, return to granting users permission to list the root-level bucket items. To list bucket content, users need permission to call the `s3:ListBucket` action, as shown in the following policy statement. To ensure that they see only the root-level content, you add a condition that users must specify an empty `prefix` in the request—that is, they are not allowed to open any of the root-level folders. Finally, you add a condition to require folder-style access by requiring user requests to include the `delimiter` parameter with the value "/".

```
{
  "Sid": "AllowRootLevelListingOfCompanyBucket",
  "Action": ["s3:ListBucket"],
  "Effect": "Allow",
  "Resource": ["arn:aws:s3:::companybucket"],
  "Condition":{
         "StringEquals":{
             "s3:prefix":[""], "s3:delimiter":["/"]
                       }
              }
}
```

When you choose a bucket on the Amazon S3 console, the console first sends the GET Bucket location request to find the AWS Region where the bucket is deployed. Then the console uses the Region-specific endpoint for the bucket to send the GET Bucket (List Objects) request. As a result, if users are going to use the console, you must grant permission for the `s3:GetBucketLocation` action as shown in the following policy statement.

```
{
   "Sid": "RequiredByS3Console",
   "Action": ["s3:GetBucketLocation"],
   "Effect": "Allow",
   "Resource": ["arn:aws:s3:::*"]
}
```

To enable users to list root-level bucket content

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

   Use your AWS account credentials, not the credentials of an IAM user, to sign in to the console.
2. Replace the existing `AllowGroupToSeeBucketListInTheConsole` managed policy that is attached to the `Consultants` group with the following policy, which also allows the

`s3:ListBucket` action. Remember to replace *`companybucket`* in the policy `Resource` with the name of your bucket.

For step-by-step instructions, see Editing IAM policies in the *IAM User Guide*. When following the step-by-step instructions, be sure to follow the steps for applying your changes to all principal entities that the policy is attached to.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
     {
       "Sid":
 "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
       "Action": [ "s3:ListAllMyBuckets", "s3:GetBucketLocation" ],
       "Effect": "Allow",
       "Resource": [ "arn:aws:s3:::*"  ]
     },
     {
       "Sid": "AllowRootLevelListingOfCompanyBucket",
       "Action": ["s3:ListBucket"],
       "Effect": "Allow",
       "Resource": ["arn:aws:s3:::companybucket"],
       "Condition":{
             "StringEquals":{
                    "s3:prefix":[""], "s3:delimiter":["/"]
                           }
                  }
     }
  ]
}
```

3. Test the updated permissions.

   a. Using the IAM user sign-in link (see To provide a sign-in link for IAM users), sign in to the AWS Management Console.

      Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

b. Choose the bucket that you created, and the console shows the root-level bucket items. If you choose any folders in the bucket, you won't be able to see the folder content because you haven't yet granted those permissions.

This test succeeds when users use the Amazon S3 console. When you choose a bucket on the console, the console implementation sends a request that includes the `prefix` parameter with an empty string as its value and the `delimiter` parameter with "`/`" as its value.

Step 4.3: Summary of the group policy

The net effect of the group policy that you added is to grant the IAM users Alice and Bob the following minimum permissions:

- List all buckets owned by the parent account.
- See root-level items in the `companybucket` bucket.

However, the users still can't do much. Next, you grant user-specific permissions, as follows:

- Allow Alice to get and put objects in the `Development` folder.
- Allow Bob to get and put objects in the `Finance` folder.

For user-specific permissions, you attach a policy to the specific user, not to the group. In the following section, you grant Alice permission to work in the `Development` folder. You can repeat the steps to grant similar permission to Bob to work in the `Finance` folder.

Step 5: Grant IAM user Alice specific permissions

Now you grant additional permissions to Alice so that she can see the content of the `Development` folder and get and put objects in that folder.

Step 5.1: Grant IAM user Alice permission to list the development folder content

For Alice to list the `Development` folder content, you must apply a policy to the user Alice that grants permission for the `s3:ListBucket` action on the `companybucket` bucket, provided the request includes the prefix `Development/`. You want this policy to be applied only to the user Alice, so you use an inline policy. For more information about inline policies, see Managed policies and inline policies in the *IAM User Guide*.

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

   Use your AWS account credentials, not the credentials of an IAM user, to sign in to the console.

2. Create an inline policy to grant the user Alice permission to list the `Development` folder content.

   a. In the navigation pane on the left, choose **Users**.

   b. Choose the username **Alice**.

   c. On the user details page, choose the **Permissions** tab and then choose **Add inline policy**.

   d. Choose the **JSON** tab.

   e. Copy the following policy, and paste it into the policy text field.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
      "Sid": "AllowListBucketIfSpecificPrefixIsIncludedInRequest",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition": { "StringLike": {"s3:prefix": ["Development/*"] }
       }
    }
  ]
}
```

   f. Choose **Review Policy**. On the next page, enter a name in the **Name** field, and then choose **Create policy**.

3. Test the change to Alice's permissions:

   a. Using the IAM user sign-in link (see To provide a sign-in link for IAM users), sign in to the AWS Management Console.

   b. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   c. On the Amazon S3 console, verify that Alice can see the list of objects in the `Development/` folder in the bucket.

When the user chooses the /Development folder to see the list of objects in it, the Amazon S3 console sends the ListObjects request to Amazon S3 with the prefix /Development. Because the user is granted permission to see the object list with the prefix Development and delimiter /, Amazon S3 returns the list of objects with the key prefix Development/, and the console displays the list.

Step 5.2: Grant IAM user Alice permissions to get and put objects in the development folder

For Alice to get and put objects in the Development folder, she needs permission to call the s3:GetObject and s3:PutObject actions. The following policy statements grant these permissions, provided that the request includes the prefix parameter with a value of Development/.

```
{
    "Sid":"AllowUserToReadWriteObjectData",
    "Action":["s3:GetObject", "s3:PutObject"],
    "Effect":"Allow",
    "Resource":["arn:aws:s3:::companybucket/Development/*"]
 }
```

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   Use your AWS account credentials, not the credentials of an IAM user, to sign in to the console.

2. Edit the inline policy that you created in the previous step.

   a. In the navigation pane on the left, choose **Users**.

   b. Choose the user name Alice.

   c. On the user details page, choose the **Permissions** tab and expand the **Inline Policies** section.

   d. Next to the name of the policy that you created in the previous step, choose **Edit Policy**.

   e. Copy the following policy and paste it into the policy text field, replacing the existing policy.

JSON

```
{
     "Version":"2012-10-17",
     "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      }
   ]
}
```

3. Test the updated policy:

   a. Using the IAM user sign-in link (see To provide a sign-in link for IAM users), sign into the AWS Management Console.

   b. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   c. On the Amazon S3 console, verify that Alice can now add an object and download an object in the `Development` folder.

Step 5.3: Explicitly deny IAM user Alice permissions to any other folders in the bucket

User Alice can now list the root-level content in the `companybucket` bucket. She can also get and put objects in the `Development` folder. If you really want to tighten the access permissions, you could explicitly deny Alice access to any other folders in the bucket. If there is any other policy (bucket policy or ACL) that grants Alice access to any other folders in the bucket, this explicit deny overrides those permissions.

You can add the following statement to the user Alice policy that requires all requests that Alice sends to Amazon S3 to include the `prefix` parameter, whose value can be either `Development/*` or an empty string.

```
{
   "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
   "Action": ["s3:ListBucket"],
   "Effect": "Deny",
   "Resource": ["arn:aws:s3:::companybucket"],
   "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                  "Null"         : {"s3:prefix":false }
    }
}
```

There are two conditional expressions in the `Condition` block. The result of these conditional expressions is combined by using the logical `AND`. If both conditions are true, the result of the combined condition is true. Because the `Effect` in this policy is `Deny`, when the `Condition` evaluates to true, users can't perform the specified `Action`.

- The `Null` conditional expression ensures that requests from Alice include the `prefix` parameter.

  The `prefix` parameter requires folder-like access. If you send a request without the `prefix` parameter, Amazon S3 returns all the object keys.

  If the request includes the `prefix` parameter with a null value, the expression evaluates to true, and so the entire `Condition` evaluates to true. You must allow an empty string as value of the `prefix` parameter. From the preceding discussion, recall that allowing the null string allows Alice to retrieve root-level bucket items as the console does in the preceding discussion. For more information, see Step 4.2: Enable users to list root-level content of a bucket.
- The `StringNotLike` conditional expression ensures that if the value of the `prefix` parameter is specified and is not `Development/*`, the request fails.

Follow the steps in the preceding section and again update the inline policy that you created for user Alice.

Copy the following policy and paste it into the policy text field, replacing the existing policy.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"AllowListBucketIfSpecificPrefixIsIncludedInRequest",
         "Action":["s3:ListBucket"],
         "Effect":"Allow",
         "Resource":["arn:aws:s3:::companybucket"],
         "Condition":{
            "StringLike":{"s3:prefix":["Development/*"]
            }
         }
      },
      {
        "Sid":"AllowUserToReadWriteObjectDataInDevelopmentFolder",
        "Action":["s3:GetObject", "s3:PutObject"],
        "Effect":"Allow",
        "Resource":["arn:aws:s3:::companybucket/Development/*"]
      },
      {
         "Sid": "ExplicitlyDenyAnyRequestsForAllOtherFoldersExceptDevelopment",
         "Action": ["s3:ListBucket"],
         "Effect": "Deny",
         "Resource": ["arn:aws:s3:::companybucket"],
         "Condition":{  "StringNotLike": {"s3:prefix":["Development/*",""] },
                        "Null"         : {"s3:prefix":false }
          }
      }
   ]
}
```

Step 6: Grant IAM user Bob specific permissions

Now you want to grant Bob permission to the `Finance` folder. Follow the steps that you used earlier to grant permissions to Alice, but replace the `Development` folder with the `Finance` folder. For step-by-step instructions, see Step 5: Grant IAM user Alice specific permissions.

Step 7: Secure the private folder

In this example, you have only two users. You granted all the minimum required permissions at the group level and granted user-level permissions only when you really need to permissions at the individual user level. This approach helps minimize the effort of managing permissions. As the number of users increases, managing permissions can become cumbersome. For example, you don't want any of the users in this example to access the content of the `Private` folder. How do you ensure that you don't accidentally grant a user permission to the `Private` folder? You add a policy that explicitly denies access to the folder. An explicit deny overrides any other permissions.

To ensure that the `Private` folder remains private, you can add the following two deny statements to the group policy:

- Add the following statement to explicitly deny any action on resources in the `Private` folder (`companybucket/Private/*`).

```
{
  "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
  "Action": ["s3:*"],
  "Effect": "Deny",
  "Resource":["arn:aws:s3:::companybucket/Private/*"]
}
```

- You also deny permission for the list objects action when the request specifies the `Private/` prefix. On the console, if Bob or Alice opens the `Private` folder, this policy causes Amazon S3 to return an error response.

```
{
  "Sid": "DenyListBucketOnPrivateFolder",
  "Action": ["s3:ListBucket"],
  "Effect": "Deny",
  "Resource": ["arn:aws:s3:::*"],
  "Condition":{
      "StringLike":{"s3:prefix":["Private/"]}
   }
}
```

Replace the `Consultants` group policy with an updated policy that includes the preceding deny statements. After the updated policy is applied, none of the users in the group can access the `Private` folder in your bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   Use your AWS account credentials, not the credentials of an IAM user, to sign in to the console.

2. Replace the existing `AllowGroupToSeeBucketListInTheConsole` managed policy that is attached to the `Consultants` group with the following policy. Remember to replace *`companybucket`* in the policy with the name of your bucket.

   For instructions, see Editing customer managed policies in the *IAM User Guide*. When following the instructions, make sure to follow the directions for applying your changes to all principal entities that the policy is attached to.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid":
 "AllowGroupToSeeBucketListAndAlsoAllowGetBucketLocationRequiredForListBucket",
      "Action": ["s3:ListAllMyBuckets", "s3:GetBucketLocation"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::*"]
    },
    {
      "Sid": "AllowRootLevelListingOfCompanyBucket",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::companybucket"],
      "Condition":{
          "StringEquals":{"s3:prefix":[""]}
       }
    },
    {
      "Sid": "RequireFolderStyleList",
      "Action": ["s3:ListBucket"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
      "Condition":{
          "StringNotEquals":{"s3:delimiter":"/"}
       }
```

```
     },
    {
      "Sid": "ExplictDenyAccessToPrivateFolderToEveryoneInTheGroup",
      "Action": ["s3:*"],
      "Effect": "Deny",
      "Resource":["arn:aws:s3:::companybucket/Private/*"]
    },
    {
      "Sid": "DenyListBucketOnPrivateFolder",
      "Action": ["s3:ListBucket"],
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::*"],
      "Condition":{
          "StringLike":{"s3:prefix":["Private/"]}
       }
    }
  ]
}
```

Step 8: Clean up

To clean up, open the IAM Console and remove the users Alice and Bob. For step-by-step instructions, see Deleting an IAM user in the *IAM User Guide*.

To ensure that you aren't charged further for storage, you should also delete the objects and the bucket that you created for this exercise.

Related resources

- Managing IAM policies in the *IAM User Guide*

Identity-based policy examples for Amazon S3

This section shows several example AWS Identity and Access Management (IAM) identity-based policies for controlling access to Amazon S3. For example *bucket policies* (resource-based policies), see Bucket policies for Amazon S3. For information about IAM policy language, see Policies and permissions in Amazon S3.

The following example policies will work if you use them programmatically. However, to use them with the Amazon S3 console, you must grant additional permissions that are required by

the console. For information about using policies such as these with the Amazon S3 console, see Controlling access to a bucket with user policies.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Topics

- Allowing an IAM user access to one of your buckets
- Allowing each IAM user access to a folder in a bucket
- Allowing a group to have a shared folder in Amazon S3
- Allowing all your users to read objects in a portion of a bucket
- Allowing a partner to drop files into a specific portion of a bucket
- Restricting access to Amazon S3 buckets within a specific AWS account
- Restricting access to Amazon S3 buckets within your organizational unit
- Restricting access to Amazon S3 buckets within your organization
- Granting permission to retrieve the PublicAccessBlock configuration for an AWS account
- Restricting bucket creation to one Region

Allowing an IAM user access to one of your buckets

In this example, you want to grant an IAM user in your AWS account access to one of your buckets, *amzn-s3-demo-bucket1*, and allow the user to add, update, and delete objects.

In addition to granting the `s3:PutObject`, `s3:GetObject`, and `s3:DeleteObject` permissions to the user, the policy also grants the `s3:ListAllMyBuckets`, `s3:GetBucketLocation`, and `s3:ListBucket` permissions. These are the additional permissions required by the console. Also, the `s3:PutObjectAcl` and the `s3:GetObjectAcl` actions are required to be able to copy, cut, and paste objects in the console. For an example walkthrough that grants permissions to users and tests them using the console, see Controlling access to a bucket with user policies.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
```

```
            "Effect":"Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource":"*"
        },
        {
            "Effect":"Allow",
            "Action":["s3:ListBucket","s3:GetBucketLocation"],
            "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1"
        },
        {
            "Effect":"Allow",
            "Action":[
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:DeleteObject"
            ],
            "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*"
        }
    ]
}
```

Allowing each IAM user access to a folder in a bucket

In this example, you want two IAM users, Mary and Carlos, to have access to your bucket, *amzn-s3-demo-bucket1*, so that they can add, update, and delete objects. However, you want to restrict each user's access to a single prefix (folder) in the bucket. You might create folders with names that match their usernames.

```
amzn-s3-demo-bucket1
   Mary/
   Carlos/
```

To grant each user access only to their folder, you can write a policy for each user and attach it individually. For example, you can attach the following policy to the user Mary to allow her specific Amazon S3 permissions on the *amzn-s3-demo-bucket1*/*Mary* folder.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/Mary/*"
      }
   ]
}
```

You can then attach a similar policy to the user Carlos, specifying the folder *`Carlos`* in the `Resource` value.

Instead of attaching policies to individual users, you can write a single policy that uses a policy variable and then attach the policy to a group. First, you must create a group and add both Mary and Carlos to the group. The following example policy allows a set of Amazon S3 permissions in the *`amzn-s3-demo-bucket1`*`/${aws:username}` folder. When the policy is evaluated, the policy variable `${aws:username}` is replaced by the requester's username. For example, if Mary sends a request to put an object, the operation is allowed only if Mary is uploading the object to the *`amzn-s3-demo-bucket1`*`/Mary` folder.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
```

```
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/${aws:username}/*"
      }
   ]
}
```

Note

When using policy variables, you must explicitly specify version 2012-10-17 in the policy. The default version of the IAM policy language, 2008-10-17, does not support policy variables.

If you want to test the preceding policy on the Amazon S3 console, the console requires additional permissions, as shown in the following policy. For information about how the console uses these permissions, see Controlling access to a bucket with user policies.

JSON

```
{
 "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowGroupToSeeBucketListInTheConsole",
      "Action": [
       "s3:ListAllMyBuckets",
       "s3:GetBucketLocation"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Sid": "AllowRootLevelListingOfTheBucket",
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1",
```

```
      "Condition": {
            "StringEquals": {
                    "s3:prefix": [""], "s3:delimiter": ["/"]
                           }
                 }
    },
    {
      "Sid": "AllowListBucketOfASpecificUserPrefix",
      "Action": "s3:ListBucket",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1",
      "Condition": {  "StringLike": {"s3:prefix": ["${aws:username}/*"] }
       }
    },
      {
     "Sid": "AllowUserSpecificActionsOnlyInTheSpecificUserPrefix",
         "Effect": "Allow",
         "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/${aws:username}/*"
      }
  ]
}
```

Note

In the 2012-10-17 version of the policy, policy variables start with $. This change in syntax can potentially create a conflict if your object key (object name) includes a $.
To avoid this conflict, specify the $ character by using `${$}`. For example, to include the object key `my$file` in a policy, specify it as `my${$}file`.

Although IAM user names are friendly, human-readable identifiers, they aren't required to be globally unique. For example, if the user Carlos leaves the organization and another Carlos joins, then the new Carlos could access the old Carlos's information.

Instead of using usernames, you could create folders based on IAM user IDs. Each IAM user ID is unique. In this case, you must modify the preceding policy to use the `${aws:userid}` policy variable. For more information about user identifiers, see IAM Identifiers in the *IAM User Guide*.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/home/${aws:userid}/*"
      }
   ]
}
```

Allowing non-IAM users (mobile app users) access to folders in a bucket

Suppose that you want to develop a mobile app, a game that stores users' data in an S3 bucket. For each app user, you want to create a folder in your bucket. You also want to limit each user's access to their own folder. But you can't create folders before someone downloads your app and starts playing the game, because you don't have their user ID.

In this case, you can require users to sign in to your app by using public identity providers such as Login with Amazon, Facebook, or Google. After users have signed in to your app through one of these providers, they have a user ID that you can use to create user-specific folders at runtime.

You can then use web identity federation in AWS Security Token Service to integrate information from the identity provider with your app and to get temporary security credentials for each user. You can then create IAM policies that allow the app to access your bucket and perform such operations as creating user-specific folders and uploading data. For more information about web identity federation, see About web identity Federation in the *IAM User Guide*.

Allowing a group to have a shared folder in Amazon S3

Attaching the following policy to the group grants everybody in the group access to the following folder in Amazon S3: *amzn-s3-demo-bucket1*/share/marketing. Group members are allowed to access only the specific Amazon S3 permissions shown in the policy and only for objects in the specified folder.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/share/marketing/*"
      }
   ]
}
```

Allowing all your users to read objects in a portion of a bucket

In this example, you create a group named *AllUsers*, which contains all the IAM users that are owned by the AWS account. You then attach a policy that gives the group access to GetObject and GetObjectVersion, but only for objects in the *amzn-s3-demo-bucket1*/readonly folder.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
```

```
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/readonly/*"
      }
   ]
}
```

Allowing a partner to drop files into a specific portion of a bucket

In this example, you create a group called *AnyCompany* that represents a partner company. You create an IAM user for the specific person or application at the partner company that needs access, and then you put the user in the group.

You then attach a policy that gives the group `PutObject` access to the following folder in a bucket:

amzn-s3-demo-bucket1/uploads/anycompany

You want to prevent the *AnyCompany* group from doing anything else with the bucket, so you add a statement that explicitly denies permission to any Amazon S3 actions except `PutObject` on any Amazon S3 resource in the AWS account.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/uploads/anycompany/*"
      },
      {
         "Effect":"Deny",
         "Action":"s3:*",
         "NotResource":"arn:aws:s3:::amzn-s3-demo-bucket1/uploads/anycompany/*"
      }
   ]
}
```

Restricting access to Amazon S3 buckets within a specific AWS account

If you want to ensure that your Amazon S3 principals are accessing only the resources that are inside of a trusted AWS account, you can restrict access. For example, this identity-based IAM policy uses a Deny effect to block access to Amazon S3 actions, unless the Amazon S3 resource that's being accessed is in account *222222222222*. To prevent an IAM principal in an AWS account from accessing Amazon S3 objects outside of the account, attach the following IAM policy:

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyS3AccessOutsideMyBoundary",
      "Effect": "Deny",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "222222222222"
          ]
        }
      }
    }
  ]
}
```

Note

This policy doesn't replace your existing IAM access controls, because it doesn't grant any access. Instead, this policy acts as an additional guardrail for your other IAM permissions, regardless of the permissions granted through other IAM policies.

Make sure to replace account ID *222222222222* in the policy with your own AWS account. To apply a policy to multiple accounts while still maintaining this restriction, replace the account ID with the `aws:PrincipalAccount` condition key. This condition requires that the principal and the resource must be in the same account.

Restricting access to Amazon S3 buckets within your organizational unit

If you have an organizational unit (OU) set up in AWS Organizations, you might want to restrict Amazon S3 bucket access to a specific part of your organization. In this example, we'll use the `aws:ResourceOrgPaths` key to restrict Amazon S3 bucket access to an OU in your organization. For this example, the OU ID is *ou-acroot-exampleou*. Make sure to replace this value in your own policy with your own OU IDs.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "DenyS3AccessOutsideMyBoundary",
       "Effect": "Deny",
       "Action": [
         "s3:*"
       ],
       "Resource": "*",
       "Condition": {
         "ForAllValues:StringNotLike": {
           "aws:ResourceOrgPaths": [
             "o-acorg/r-acroot/ou-acroot-exampleou/*"
           ]
         }
       }
     }
   ]
 }
```

> **Note**
>
> This policy doesn't grant any access. Instead, this policy acts as a backstop for your other IAM permissions, preventing your principals from accessing Amazon S3 objects outside of an OU-defined boundary.

The policy denies access to Amazon S3 actions unless the Amazon S3 object that's being accessed is in the *ou-acroot-exampleou* OU in your organization. The IAM policy condition requires `aws:ResourceOrgPaths`, a multivalued condition key, to contain any of the listed OU paths. The policy uses the `ForAllValues:StringNotLike` operator to compare the values of `aws:ResourceOrgPaths` to the listed OUs without case-sensitive matching.

Restricting access to Amazon S3 buckets within your organization

To restrict access to Amazon S3 objects within your organization, attach an IAM policy to the root of the organization, applying it to all accounts in your organization. To require your IAM principals to follow this rule, use a service-control policy (SCP). If you choose to use an SCP, make sure to thoroughly test the SCP before attaching the policy to the root of the organization.

In the following example policy, access is denied to Amazon S3 actions unless the Amazon S3 object that's being accessed is in the same organization as the IAM principal that is accessing it:

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "DenyS3AccessOutsideMyBoundary",
       "Effect": "Deny",
       "Action": [
         "s3:*"
       ],
       "Resource": "arn:aws:s3:::*/*",
       "Condition": {
         "StringNotEquals": {
           "aws:ResourceOrgID": "${aws:PrincipalOrgID}"
         }
       }
```

```
      }
    ]
 }
```

> **Note**
>
> This policy doesn't grant any access. Instead, this policy acts as a backstop for your other IAM permissions, preventing your principals from accessing any Amazon S3 objects outside of your organization. This policy also applies to Amazon S3 resources that are created after the policy is put into effect.

The IAM policy condition in this example requires `aws:ResourceOrgID` and `aws:PrincipalOrgID` to be equal to each other. With this requirement, the principal making the request and the resource being accessed must be in the same organization.

Granting permission to retrieve the PublicAccessBlock configuration for an AWS account

The following example identity-based policy grants the `s3:GetAccountPublicAccessBlock` permission to a user. For these permissions, you set the `Resource` value to `"*"`. For information about resource ARNs, see Policy resources for Amazon S3.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action":[
            "s3:GetAccountPublicAccessBlock"
         ],
         "Resource":[
            "*"
         ]
       }
    ]
}
```

Restricting bucket creation to one Region

Suppose that an AWS account administrator wants to grant its user (Dave) permission to create a bucket in the South America (São Paulo) Region only. The account administrator can attach the following user policy granting the `s3:CreateBucket` permission with a condition as shown. The key-value pair in the `Condition` block specifies the `s3:LocationConstraint` key and the `sa-east-1` Region as its value.

> **Note**
>
> In this example, the bucket owner is granting permission to one of its users, so either a bucket policy or a user policy can be used. This example shows a user policy.

For a list of Amazon S3 Regions, see Regions and Endpoints in the *AWS General Reference*.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "statement1",
         "Effect": "Allow",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       }
    ]
}
```

Add explicit deny

The preceding policy restricts the user from creating a bucket in any other Region except `sa-east-1`. However, some other policy might grant this user permission to create buckets in another Region. For example, if the user belongs to a group, the group might have a policy attached to it

that allows all users in the group permission to create buckets in another Region. To ensure that the user doesn't get permission to create buckets in any other Region, you can add an explicit deny statement in the above policy.

The `Deny` statement uses the `StringNotLike` condition. That is, a create bucket request is denied if the location constraint is not `sa-east-1`. The explicit deny doesn't allow the user to create a bucket in any other Region, no matter what other permission the user gets. The following policy includes an explicit deny statement.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"statement1",
         "Effect":"Allow",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       },
      {
         "Sid":"statement2",
         "Effect":"Deny",
         "Action": "s3:CreateBucket",
         "Resource": "arn:aws:s3:::*",
         "Condition": {
             "StringNotLike": {
                 "s3:LocationConstraint": "sa-east-1"
             }
         }
       }
    ]
}
```

Test the policy with the AWS CLI

You can test the policy using the following `create-bucket` AWS CLI command. This example uses the `bucketconfig.txt` file to specify the location constraint. Note the Windows file path. You need to update the bucket name and path as appropriate. You must provide user credentials using the `--profile` parameter. For more information about setting up and using the AWS CLI, see Developing with Amazon S3 using the AWS CLI in the *Amazon S3 API Reference*.

```
aws s3api create-bucket --bucket examplebucket --profile AccountADave --create-bucket-
configuration file://c:/Users/someUser/bucketconfig.txt
```

The `bucketconfig.txt` file specifies the configuration as follows.

```
{"LocationConstraint": "sa-east-1"}
```

Walkthroughs that use policies to manage access to your Amazon S3 resources

This topic provides the following introductory walkthrough examples for granting access to Amazon S3 resources. These examples use the AWS Management Console to create resources (buckets, objects, users) and grant them permissions. The examples then show you how to verify permissions using the command line tools, so you don't have to write any code. We provide commands using both the AWS Command Line Interface (AWS CLI) and the AWS Tools for Windows PowerShell.

- Example 1: Bucket owner granting its users bucket permissions

  The IAM users you create in your account have no permissions by default. In this exercise, you grant a user permission to perform bucket and object operations.
- Example 2: Bucket owner granting cross-account bucket permissions

  In this exercise, a bucket owner, Account A, grants cross-account permissions to another AWS account, Account B. Account B then delegates those permissions to users in its account.
- **Managing object permissions when the object and bucket owners are not the same**

  The example scenarios in this case are about a bucket owner granting object permissions to others, but not all objects in the bucket are owned by the bucket owner. What permissions does the bucket owner need, and how can it delegate those permissions?

  The AWS account that creates a bucket is called the *bucket owner*. The owner can grant other AWS accounts permission to upload objects, and the AWS accounts that create objects own them. The bucket owner has no permissions on those objects created by other AWS accounts. If the bucket owner writes a bucket policy granting access to objects, the policy doesn't apply to objects that are owned by other accounts.

  In this case, the object owner must first grant permissions to the bucket owner using an object ACL. The bucket owner can then delegate those object permissions to others, to users in its own account, or to another AWS account, as illustrated by the following examples.

  - Example 3: Bucket owner granting permissions to objects it does not own

    In this exercise, the bucket owner first gets permissions from the object owner. The bucket owner then delegates those permissions to users in its own account.
  - Example 4 - Bucket owner granting cross-account permission to objects it does not own

After receiving permissions from the object owner, the bucket owner can't delegate permission to other AWS accounts because cross-account delegation isn't supported (see Permission delegation). Instead, the bucket owner can create an IAM role with permissions to perform specific operations (such as get object) and allow another AWS account to assume that role. Anyone who assumes the role can then access objects. This example shows how a bucket owner can use an IAM role to enable this cross-account delegation.

Before you try the example walkthroughs

These examples use the AWS Management Console to create resources and grant permissions. To test permissions, the examples use the command line tools, AWS CLI, and AWS Tools for Windows PowerShell, so you don't need to write any code. To test permissions, you must set up one of these tools. For more information, see Setting up the tools for the walkthroughs.

In addition, when creating resources, these examples don't use root user credentials of an AWS account. Instead, you create an administrator user in these accounts to perform these tasks.

About using an administrator user to create resources and grant permissions

AWS Identity and Access Management (IAM) recommends not using the root user credentials of your AWS account to make requests. Instead, create an IAM user or role, grant them full access, and then use their credentials to make requests. We refer to this as an administrative user or role. For more information, go to AWS account root user credentials and IAM identities in the *AWS General Reference* and IAM Best Practices in the *IAM User Guide*.

All example walkthroughs in this section use the administrator user credentials. If you have not created an administrator user for your AWS account, the topics show you how.

To sign in to the AWS Management Console using the user credentials, you must use the IAM user sign-In URL. The IAM Console provides this URL for your AWS account. The topics show you how to get the URL.

Setting up the tools for the walkthroughs

The introductory examples (see Walkthroughs that use policies to manage access to your Amazon S3 resources) use the AWS Management Console to create resources and grant permissions. To test permissions, the examples use the command line tools, AWS Command Line Interface (AWS CLI) and AWS Tools for Windows PowerShell, so you don't need to write any code. To test permissions, you must set up one of these tools.

To set up the AWS CLI

1. Download and configure the AWS CLI. For instructions, see the following topics in the *AWS Command Line Interface User Guide*:

   Install or update to the latest version of the AWS Command Line Interface

   Get started with the AWS Command Line Interface

2. Set the default profile.

   You store user credentials in the AWS CLI config file. Create a default profile in the config file using your AWS account credentials. For instructions on finding and editing your AWS CLI config file, see Configuration and credential file settings.

   ```
   [default]
   aws_access_key_id = access key ID
   aws_secret_access_key = secret access key
   region = us-west-2
   ```

3. Verify the setup by entering the following command at the command prompt. Both these commands don't provide credentials explicitly, so the credentials of the default profile are used.

   - Try the `help` command.

     ```
     aws help
     ```

   - To get a list of buckets on the configured account, use the `aws s3 ls` command.

     ```
     aws s3 ls
     ```

As you go through the walkthroughs, you will create users, and you will save user credentials in the config files by creating profiles, as the following example shows. These profiles have the names of `AccountAadmin` and `AccountBadmin`.

```
[profile AccountAadmin]
aws_access_key_id = User AccountAadmin access key ID
aws_secret_access_key = User AccountAadmin secret access key
region = us-west-2
```

```
[profile AccountBadmin]
aws_access_key_id = Account B access key ID
aws_secret_access_key = Account B secret access key
region = us-east-1
```

To run a command using these user credentials, you add the `--profile` parameter specifying the profile name. The following AWS CLI command retrieves a listing of objects in *examplebucket* and specifies the `AccountBadmin` profile.

```
aws s3 ls s3://examplebucket --profile AccountBadmin
```

Alternatively, you can configure one set of user credentials as the default profile by changing the `AWS_DEFAULT_PROFILE` environment variable from the command prompt. After you've done this, whenever you perform AWS CLI commands without the `--profile` parameter, the AWS CLI uses the profile you set in the environment variable as the default profile.

```
$ export AWS_DEFAULT_PROFILE=AccountAadmin
```

To set up AWS Tools for Windows PowerShell

1. Download and configure the AWS Tools for Windows PowerShell. For instructions, go to Installing the AWS Tools for Windows PowerShell in the *AWS Tools for PowerShell User Guide*.

   > **Note**
   >
   > To load the AWS Tools for Windows PowerShell module, you must enable PowerShell script execution. For more information, see Enable Script Execution in the *AWS Tools for PowerShell User Guide*.

2. For these walkthroughs, you specify AWS credentials per session using the `Set-AWSCredentials` command. The command saves the credentials to a persistent store (`-StoreAs` parameter).

   ```
   Set-AWSCredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -
   storeas string
   ```

3. Verify the setup.

- To retrieve a list of available commands that you can use for Amazon S3 operations, run the `Get-Command` command.

```
Get-Command -module awspowershell -noun s3* -StoredCredentials string
```

- To retrieve a list of objects in a bucket, run the `Get-S3Object` command.

```
Get-S3Object -BucketName bucketname -StoredCredentials string
```

For a list of commands, see AWS Tools for PowerShell Cmdlet Reference.

Now you're ready to try the walkthroughs. Follow the links provided at the beginning of each section.

Example 1: Bucket owner granting its users bucket permissions

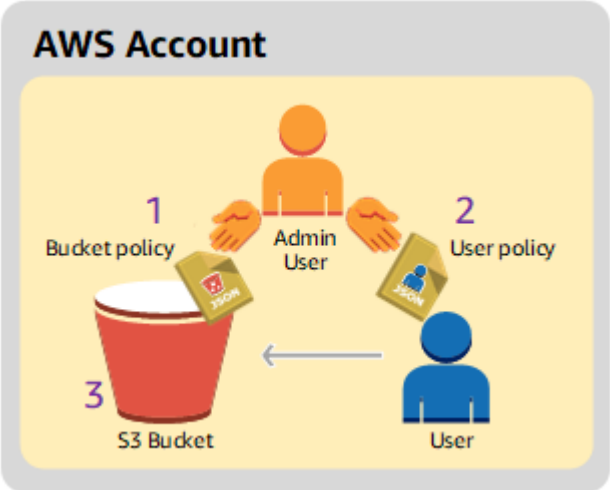
> **Important**
>
> Granting permissions to IAM roles is a better practice than granting permissions to individual users.For more information about how to grant permissions to IAM roles, see Understanding cross-account permissions and using IAM roles.

Topics

- Preparing for the walkthrough
- Step 1: Create resources in Account A and grant permissions
- Step 2: Test permissions

In this walkthrough, an AWS account owns a bucket, and the account includes an IAM user By default, the user has no permissions. For the user to perform any tasks, the parent account must grant them permissions. The bucket owner and parent account are the same. Therefore, to grant the user permissions on the bucket, the AWS account can use a bucket policy, a user policy, or both. The account owner will grant some permissions using a bucket policy and other permissions using a user policy.

The following steps summarize the walkthrough:



1. Account administrator creates a bucket policy granting a set of permissions to the user.
2. Account administrator attaches a user policy to the user granting additional permissions.
3. User then tries permissions granted through both the bucket policy and the user policy.

For this example, you will need an AWS account. Instead of using the root user credentials of the account, you will create an administrator user (see About using an administrator user to create resources and grant permissions). We refer to the AWS account and the administrator user as shown in the following table.

Account ID	Account referred to as	Administrator user in the account
1111-1111-1111	Account A	AccountAadmin

> **Note**
>
> The administrator user in this example is **AccountAadmin**, which refers to Account A, and not **AccountAdmin**.

All the tasks of creating users and granting permissions are done in the AWS Management Console. To verify permissions, the walkthrough uses the command line tools, AWS Command Line Interface (AWS CLI) and AWS Tools for Windows PowerShell, so you don't need to write any code.

Preparing for the walkthrough

1. Make sure you have an AWS account and that it has a user with administrator privileges.

   a. Sign up for an AWS account, if needed. We refer to this account as Account A.

      i. Go to https://aws.amazon.com/s3 and choose **Create an AWS account**.

      ii. Follow the on-screen instructions.

      AWS will notify you by email when your account is active and available for you to use.

   b. In Account A, create an administrator user **`AccountAadmin`**. Using Account A credentials, sign in to the IAM console and do the following:

      i. Create user **`AccountAadmin`** and note the user security credentials.

      For instructions, see Creating an IAM user in your AWS account in the *IAM User Guide*.

      ii. Grant administrator privileges to **AccountAadmin** by attaching a user policy giving full access.

For instructions, see Managing IAM policies in the *IAM User Guide*.

iii. Note the **IAM user Sign-In URL** for **AccountAadmin**. You will need to use this URL when signing in to the AWS Management Console. For more information about where to find the sign-in URL, see Sign in to the AWS Management Console as an IAM user in *IAM User Guide*. Note the URL for each of the accounts.

2. Set up either the AWS CLI or the AWS Tools for Windows PowerShell. Make sure that you save administrator user credentials as follows:

   - If using the AWS CLI, create a profile, `AccountAadmin`, in the config file.
   - If using the AWS Tools for Windows PowerShell, make sure you store credentials for the session as `AccountAadmin`.

   For instructions, see Setting up the tools for the walkthroughs.

Step 1: Create resources in Account A and grant permissions

Using the credentials of user `AccountAadmin` in Account A, and the special IAM user sign-in URL, sign in to the AWS Management Console and do the following:

1. Create resources of a bucket and an IAM user

   a. In the Amazon S3 console, create a bucket. Note the AWS Region in which you created the bucket. For instructions, see Creating a general purpose bucket.

   b. In the IAM Console, do the following:

      i. Create a user named Dave.

      For step-by-step instructions, see Creating IAM users (console) in the *IAM User Guide*.

      ii. Note the `UserDave` credentials.

      iii. Note the Amazon Resource Name (ARN) for user Dave. In the IAM Console, select the user, and the **Summary** tab provides the user ARN.

2. Grant permissions.

   Because the bucket owner and the parent account to which the user belongs are the same, the AWS account can grant user permissions using a bucket policy, a user policy, or both. In this

example, you do both. If the object is also owned by the same account, the bucket owner can grant object permissions in the bucket policy (or an IAM policy).

a. In the Amazon S3 console, attach the following bucket policy to *awsexamplebucket1*.

The policy has two statements.

- The first statement grants Dave the bucket operation permissions `s3:GetBucketLocation` and `s3:ListBucket`.
- The second statement grants the `s3:GetObject` permission. Because Account A also owns the object, the account administrator is able to grant the `s3:GetObject` permission.

In the `Principal` statement, Dave is identified by his user ARN. For more information about policy elements, see Policies and permissions in Amazon S3.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Dave"
            },
            "Action": [
                "s3:GetBucketLocation",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::awsexamplebucket1"
            ]
        },
        {
            "Sid": "statement2",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Dave"
```

```
            },
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::awsexamplebucket1/*"
            ]
        }
    ]
}
```

b. Create an inline policy for the user Dave by using the following policy. The policy grants Dave the `s3:PutObject` permission. You need to update the policy by providing your bucket name.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "PermissionForObjectOperations",
         "Effect": "Allow",
         "Action": [
            "s3:PutObject"
         ],
         "Resource": [
            "arn:aws:s3:::awsexamplebucket1/*"
         ]
      }
   ]
}
```

For instructions, see Managing IAMpolicies in the *IAM User Guide*. Note you need to sign in to the console using Account A credentials.

Step 2: Test permissions

Using Dave's credentials, verify that the permissions work. You can use either of the following two procedures.

Test permissions using the AWS CLI

1. Update the AWS CLI config file by adding the following `UserDaveAccountA` profile. For more information, see Setting up the tools for the walkthroughs.

   ```
   [profile UserDaveAccountA]
   aws_access_key_id = access-key
   aws_secret_access_key = secret-access-key
   region = us-east-1
   ```

2. Verify that Dave can perform the operations as granted in the user policy. Upload a sample object using the following AWS CLI `put-object` command.

   The `--body` parameter in the command identifies the source file to upload. For example, if the file is in the root of the C: drive on a Windows machine, you specify `c:\HappyFace.jpg`. The `--key` parameter provides the key name for the object.

   ```
   aws s3api put-object --bucket awsexamplebucket1 --key HappyFace.jpg --
   body HappyFace.jpg --profile UserDaveAccountA
   ```

   Run the following AWS CLI command to get the object.

   ```
   aws s3api get-object --bucket awsexamplebucket1 --key HappyFace.jpg OutputFile.jpg
    --profile UserDaveAccountA
   ```

Test permissions using the AWS Tools for Windows PowerShell

1. Store Dave's credentials as `AccountADave`. You then use these credentials to PUT and GET an object.

   ```
   set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
    AccountADave
   ```

2. Upload a sample object using the AWS Tools for Windows PowerShell `Write-S3Object` command using user Dave's stored credentials.

   ```
   Write-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file HappyFace.jpg
    -StoredCredentials AccountADave
   ```

Download the previously uploaded object.

```
Read-S3Object -bucketname awsexamplebucket1 -key HappyFace.jpg -file Output.jpg -
StoredCredentials AccountADave
```

Example 2: Bucket owner granting cross-account bucket permissions

> **Important**
>
> Granting permissions to IAM roles is a better practice than granting permissions to individual users. To learn how to do this, see Understanding cross-account permissions and using IAM roles.

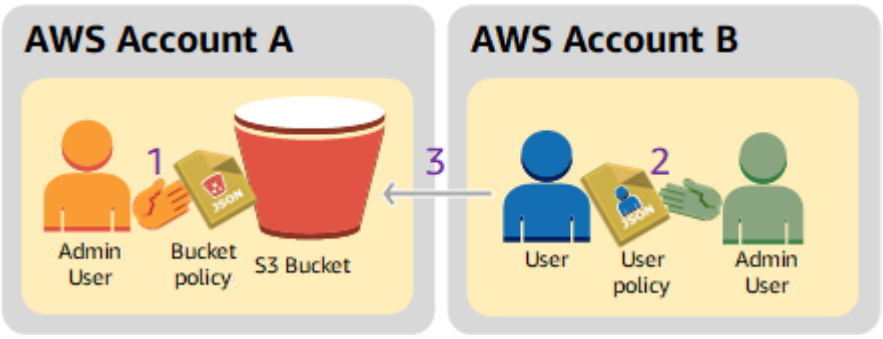
Topics

- Preparing for the walkthrough
- Step 1: Do the Account A tasks
- Step 2: Do the Account B tasks
- Step 3: (Optional) Try explicit deny
- Step 4: Clean up

An AWS account—for example, Account A—can grant another AWS account, Account B, permission to access its resources such as buckets and objects. Account B can then delegate those permissions to users in its account. In this example scenario, a bucket owner grants cross-account permission to another account to perform specific bucket operations.

> **Note**
>
> Account A can also directly grant a user in Account B permissions using a bucket policy. However, the user will still need permission from the parent account, Account B, to which the user belongs, even if Account B doesn't have permissions from Account A. As long as the user has permission from both the resource owner and the parent account, the user will be able to access the resource.

The following is a summary of the walkthrough steps:



1. Account A administrator user attaches a bucket policy granting cross-account permissions to Account B to perform specific bucket operations.

   Note that administrator user in Account B will automatically inherit the permissions.
2. Account B administrator user attaches user policy to the user delegating the permissions it received from Account A.
3. User in Account B then verifies permissions by accessing an object in the bucket owned by Account A.

For this example, you need two accounts. The following table shows how we refer to these accounts and the administrator users in them. In accordance with the IAM guidelines (see About using an administrator user to create resources and grant permissions), we don't use the root user credentials in this walkthrough. Instead, you create an administrator user in each account and use those credentials when creating resources and granting them permissions.

AWS account ID	Account referred to as	Administrator user in the account
1111-1111-1111	Account A	AccountAadmin
2222-2222-2222	Account B	AccountBadmin

All the tasks of creating users and granting permissions are done in the AWS Management Console. To verify permissions, the walkthrough uses the command line tools, AWS Command Line Interface (CLI) and AWS Tools for Windows PowerShell, so you don't need to write any code.

Preparing for the walkthrough

1. Make sure you have two AWS accounts and that each account has one administrator user as shown in the table in the preceding section.

   a. Sign up for an AWS account, if needed.

   b. Using Account A credentials, sign in to the IAM console to create the administrator user:

      i. Create user **`AccountAadmin`** and note the security credentials. For instructions, see Creating an IAM user in Your AWS account in the *IAM User Guide*.

      ii. Grant administrator privileges to **AccountAadmin** by attaching a user policy giving full access. For instructions, see Working with Policies in the *IAM User Guide*.

   c. While you are in the IAM console, note the **IAM user Sign-In URL** on the **Dashboard**. All users in the account must use this URL when signing in to the AWS Management Console.

      For more information, see How Users Sign in to Your Account in *IAM User Guide*.

   d. Repeat the preceding step using Account B credentials and create administrator user **`AccountBadmin`**.

2. Set up either the AWS Command Line Interface (AWS CLI) or the AWS Tools for Windows PowerShell. Make sure that you save administrator user credentials as follows:

   - If using the AWS CLI, create two profiles, `AccountAadmin` and `AccountBadmin`, in the config file.
   - If using the AWS Tools for Windows PowerShell, make sure that you store credentials for the session as `AccountAadmin` and `AccountBadmin`.

   For instructions, see Setting up the tools for the walkthroughs.

3. Save the administrator user credentials, also referred to as profiles. You can use the profile name instead of specifying credentials for each command you enter. For more information, see Setting up the tools for the walkthroughs.

   a. Add profiles in the AWS CLI credentials file for each of the administrator users, `AccountAadmin` and `AccountBadmin`, in the two accounts.

```
[AccountAadmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
```

```
region = us-east-1

[AccountBadmin]
aws_access_key_id = access-key-ID
aws_secret_access_key = secret-access-key
region = us-east-1
```

b. If you're using the AWS Tools for Windows PowerShell, run the following command.

```
set-awscredentials -AccessKey AcctA-access-key-ID -SecretKey AcctA-secret-
access-key -storeas AccountAadmin
set-awscredentials -AccessKey AcctB-access-key-ID -SecretKey AcctB-secret-
access-key -storeas AccountBadmin
```

Step 1: Do the Account A tasks

Step 1.1: Sign in to the AWS Management Console

Using the IAM user sign-in URL for Account A, first sign in to the AWS Management Console as **AccountAadmin** user. This user will create a bucket and attach a policy to it.

Step 1.2: Create a bucket

1. In the Amazon S3 console, create a bucket. This exercise assumes the bucket is created in the US East (N. Virginia) AWS Region and is named *amzn-s3-demo-bucket*.

   For instructions, see Creating a general purpose bucket.
2. Upload a sample object to the bucket.

   For instructions, go to Step 2: Upload an object to your bucket.

Step 1.3: Attach a bucket policy to grant cross-account permissions to Account B

The bucket policy grants the `s3:GetLifecycleConfiguration` and `s3:ListBucket` permissions to Account B. It's assumed that you're still signed in to the console using **AccountAadmin** user credentials.

1. Attach the following bucket policy to *amzn-s3-demo-bucket*. The policy grants Account B permission for the `s3:GetLifecycleConfiguration` and `s3:ListBucket` actions.

   For instructions, see Adding a bucket policy by using the Amazon S3 console.

JSON

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Example permissions",
         "Effect": "Allow",
         "Principal": {
            "AWS": "arn:aws:iam::2222-2222-2222:root"
         },
         "Action": [
            "s3:GetLifecycleConfiguration",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      }
   ]
}
```

2. Verify that Account B (and thus its administrator user) can perform the operations.

   - Verify using the AWS CLI

   ```
   aws s3 ls s3://amzn-s3-demo-bucket --profile AccountBadmin
   aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --
   profile AccountBadmin
   ```

   - Verify using the AWS Tools for Windows PowerShell

   ```
   get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBadmin
   get-s3bucketlifecycleconfiguration -BucketName amzn-s3-demo-bucket -
   StoredCredentials AccountBadmin
   ```

Step 2: Do the Account B tasks

Now the Account B administrator creates a user, Dave, and delegates the permissions received from Account A.

Step 2.1: Sign in to the AWS Management Console

Using the IAM user sign-in URL for Account B, first sign in to the AWS Management Console as **AccountBadmin** user.

Step 2.2: Create user Dave in Account B

In the IAM Console, create a user, **Dave**.

For instructions, see Creating IAM users (console) in the *IAM User Guide*.

Step 2.3: Delegate permissions to user Dave

Create an inline policy for the user Dave by using the following policy. You will need to update the policy by providing your bucket name.

It's assumed that you're signed in to the console using **AccountBadmin** user credentials.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "Example",
         "Effect": "Allow",
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
         ]
      }
   ]
}
```

For instructions, see Managing IAM policies in the *IAM User Guide*.

Step 2.4: Test permissions

Now Dave in Account B can list the contents of *amzn-s3-demo-bucket* owned by Account A. You can verify the permissions using either of the following procedures.

Test permissions using the AWS CLI

1. Add the UserDave profile to the AWS CLI config file. For more information about the config file, see Setting up the tools for the walkthroughs.

```
[profile UserDave]
aws_access_key_id = access-key
aws_secret_access_key = secret-access-key
region = us-east-1
```

2. At the command prompt, enter the following AWS CLI command to verify Dave can now get an object list from the *amzn-s3-demo-bucket* owned by Account A. Note the command specifies the UserDave profile.

```
aws s3 ls s3://amzn-s3-demo-bucket --profile UserDave
```

Dave doesn't have any other permissions. So, if he tries any other operation—for example, the following get-bucket-lifecycle configuration—Amazon S3 returns permission denied.

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --profile
 UserDave
```

Test permissions using AWS Tools for Windows PowerShell

1. Store Dave's credentials as AccountBDave.

```
set-awscredentials -AccessKey AccessKeyID -SecretKey SecretAccessKey -storeas
 AccountBDave
```

2. Try the List Bucket command.

```
get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBDave
```

Dave doesn't have any other permissions. So, if he tries any other operation—for example, the following get-s3bucketlifecycleconfiguration—Amazon S3 returns permission denied.

```
get-s3bucketlifecycleconfiguration -BucketName amzn-s3-demo-bucket -
StoredCredentials AccountBDave
```

Step 3: (Optional) Try explicit deny

You can have permissions granted by using an access control list (ACL), a bucket policy, or a user policy. But if there is an explicit deny set by either a bucket policy or a user policy, the explicit deny takes precedence over any other permissions. For testing, update the bucket policy and explicitly deny Account B the `s3:ListBucket` permission. The policy also grants `s3:ListBucket` permission. However, explicit deny takes precedence, and Account B or users in Account B will not be able to list objects in *amzn-s3-demo-bucket*.

1. Using credentials of user `AccountAadmin` in Account A, replace the bucket policy by the following.
2. Now if you try to get a bucket list using `AccountBadmin` credentials, access is denied.
   - Using the AWS CLI, run the following command:

     ```
     aws s3 ls s3://amzn-s3-demo-bucket --profile AccountBadmin
     ```

   - Using the AWS Tools for Windows PowerShell, run the following command:

     ```
     get-s3object -BucketName amzn-s3-demo-bucket -StoredCredentials AccountBDave
     ```

Step 4: Clean up

1. After you're done testing, you can do the following to clean up:
   - Sign in to the AWS Management Console (AWS Management Console) using Account A credentials, and do the following:
     - In the Amazon S3 console, remove the bucket policy attached to *amzn-s3-demo-bucket*. In the bucket **Properties**, delete the policy in the **Permissions** section.
     - If the bucket is created for this exercise, in the Amazon S3 console, delete the objects and then delete the bucket.
     - In the IAM Console, remove the `AccountAadmin` user.

2. Sign in to the IAM Console using Account B credentials. Delete user `AccountBadmin`. For step-by-step instructions, see Deleting an IAM user in the *IAM User Guide*.

Example 3: Bucket owner granting permissions to objects it does not own

> **Important**
>
> Granting permissions to IAM roles is a better practice than granting permissions to individual users. To learn how to do this, see Understanding cross-account permissions and using IAM roles.

Topics

- Step 0: Preparing for the walkthrough
- Step 1: Do the Account A tasks
- Step 2: Do the Account B tasks
- Step 3: Test permissions
- Step 4: Clean up

The scenario for this example is that a bucket owner wants to grant permission to access objects, but the bucket owner doesn't own all objects in the bucket. For this example, the bucket owner is trying to grant permission to users in its own account.

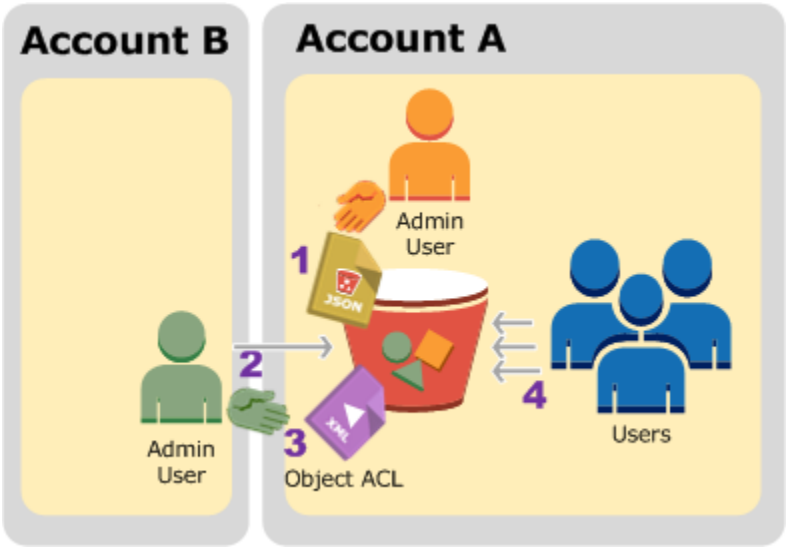
A bucket owner can enable other AWS accounts to upload objects. By default, the bucket owner doesn't own objects written to a bucket by another AWS account. Objects are owned by the accounts that write them to an S3 bucket. If the bucket owner doesn't own objects in the bucket, the object owner must first grant permission to the bucket owner using an object access control list (ACL). Then, the bucket owner can grant permissions to an object that they don't own. For more information, see Amazon S3 bucket and object ownership.

If the bucket owner applies the bucket owner enforced setting for S3 Object Ownership for the bucket, the bucket owner will own all objects in the bucket, including objects written by another AWS account. This approach resolves the issue that objects are not owned by the bucket owner. Then, you can delegate permission to users in your own account or to other AWS accounts.

Note

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.
A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

In this example, we assume the bucket owner has not applied the bucket owner enforced setting for Object Ownership. The bucket owner delegates permission to users in its own account. The following is a summary of the walkthrough steps:



1. Account A administrator user attaches a bucket policy with two statements.
   - Allow cross-account permission to Account B to upload objects.
   - Allow a user in its own account to access objects in the bucket.
2. Account B administrator user uploads objects to the bucket owned by Account A.

3. Account B administrator updates the object ACL adding grant that gives the bucket owner full-control permission on the object.
4. User in Account A verifies by accessing objects in the bucket, regardless of who owns them.

For this example, you need two accounts. The following table shows how we refer to these accounts and the administrator users in these accounts. In this walkthrough, you don't use the account root user credentials, according to the recommended IAM guidelines. For more information, see About using an administrator user to create resources and grant permissions. Instead, you create an administrator in each account and use those credentials in creating resources and granting them permissions.

AWS account ID	Account referred to as	Administrator in the account
1111-1111-1111	Account A	AccountAadmin
2222-2222-2222	Account B	AccountBadmin

All the tasks of creating users and granting permissions are done in the AWS Management Console. To verify permissions, the walkthrough uses the command line tools, AWS Command Line Interface (AWS CLI) and AWS Tools for Windows PowerShell, so you don't need to write any code.

Step 0: Preparing for the walkthrough

1. Make sure that you have two AWS accounts and each account has one administrator as shown in the table in the preceding section.
   a. Sign up for an AWS account, if needed.
   b. Using Account A credentials, sign in to the IAM Console and do the following to create an administrator user:
      - Create user **`AccountAadmin`** and note the user's security credentials. For more information about adding users, see Creating an IAM user in your AWS account in the *IAM User Guide*.
      - Grant administrator permissions to **AccountAadmin** by attaching a user policy that gives full access. For instructions, see Managing IAM policies in the *IAM User Guide*.

- In the IAM Console **Dashboard**, note the **IAM User Sign-In URL**. Users in this account must use this URL when signing in to the AWS Management Console. For more information, see How users sign in to your account in *IAM User Guide*.

c. Repeat the preceding step using Account B credentials and create administrator user **`AccountBadmin`**.

2. Set up either the AWS CLI or the Tools for Windows PowerShell. Make sure that you save the administrator credentials as follows:

   - If using the AWS CLI, create two profiles, `AccountAadmin` and `AccountBadmin`, in the config file.
   - If using the Tools for Windows PowerShell, make sure that you store credentials for the session as `AccountAadmin` and `AccountBadmin`.

   For instructions, see Setting up the tools for the walkthroughs.

Step 1: Do the Account A tasks

Perform the following steps for Account A:

Step 1.1: Sign in to the console

Using the IAM user sign-in URL for Account A, sign in to the AWS Management Console as **`AccountAadmin`** user. This user will create a bucket and attach a policy to it.

Step 1.2: Create a bucket and user, and add a bucket policy to grant user permissions

1. In the Amazon S3 console, create a bucket. This exercise assumes that the bucket is created in the US East (N. Virginia) AWS Region, and the name is *`amzn-s3-demo-bucket1`*.

   For instructions, see Creating a general purpose bucket.

2. In the IAM Console, create a user **`Dave`**.

   For step-by-step instructions, see Creating IAM users (console) in the *IAM User Guide*.

3. Note the user Dave credentials.
4. In the Amazon S3 console, attach the following bucket policy to *`amzn-s3-demo-bucket1`* bucket. For instructions, see Adding a bucket policy by using the Amazon S3 console. Follow the steps to add a bucket policy. For information about how to find account IDs, see Finding your AWS account ID.

The policy grants Account B the `s3:PutObject` and `s3:ListBucket` permissions. The policy also grants user `Dave` the `s3:GetObject` permission.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": [
                "s3:PutObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        },
        {
            "Sid": "Statement3",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Dave"
            },
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
        }
    ]
}
```

Step 2: Do the Account B tasks

Now that Account B has permissions to perform operations on Account A's bucket, the Account B administrator does the following:

- Uploads an object to Account A's bucket
- Adds a grant in the object ACL to allow Account A, the bucket owner, full control

Using the AWS CLI

1. Using the `put-object` AWS CLI command, upload an object. The `--body` parameter in the command identifies the source file to upload. For example, if the file is on the `C:` drive of a Windows machine, specify `c:\HappyFace.jpg`. The `--key` parameter provides the key name for the object.

   ```
   aws s3api put-object --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --body
    HappyFace.jpg --profile AccountBadmin
   ```

2. Add a grant to the object ACL to allow the bucket owner full control of the object. For information about how to find a canonical user ID, see Find the canonical user ID for your AWS account in the *AWS Account Management Reference Guide*.

   ```
   aws s3api put-object-acl --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --grant-
   full-control id="AccountA-CanonicalUserID" --profile AccountBadmin
   ```

Using the Tools for Windows PowerShell

1. Using the `Write-S3Object` command, upload an object.

   ```
   Write-S3Object -BucketName amzn-s3-demo-bucket1 -key HappyFace.jpg -file
    HappyFace.jpg -StoredCredentials AccountBadmin
   ```

2. Add a grant to the object ACL to allow the bucket owner full control of the object.

   ```
   Set-S3ACL -BucketName amzn-s3-demo-bucket1 -Key HappyFace.jpg -CannedACLName
    "bucket-owner-full-control" -StoredCreden
   ```

Step 3: Test permissions

Now verify that user Dave in Account A can access the object owned by Account B.

Using the AWS CLI

1. Add user Dave credentials to the AWS CLI config file and create a new profile, `UserDaveAccountA`. For more information, see Setting up the tools for the walkthroughs.

   ```
   [profile UserDaveAccountA]
   aws_access_key_id = access-key
   aws_secret_access_key = secret-access-key
   region = us-east-1
   ```

2. Run the `get-object` CLI command to download `HappyFace.jpg` and save it locally. You provide user Dave credentials by adding the `--profile` parameter.

   ```
   aws s3api get-object --bucket amzn-s3-demo-bucket1 --key
    HappyFace.jpg Outputfile.jpg --profile UserDaveAccountA
   ```

Using the Tools for Windows PowerShell

1. Store user Dave AWS credentials, as `UserDaveAccountA`, to persistent store.

   ```
   Set-AWSCredentials -AccessKey UserDave-AccessKey -SecretKey UserDave-
   SecretAccessKey -storeas UserDaveAccountA
   ```

2. Run the `Read-S3Object` command to download the `HappyFace.jpg` object and save it locally. You provide user Dave credentials by adding the `-StoredCredentials` parameter.

   ```
   Read-S3Object -BucketName amzn-s3-demo-bucket1 -Key HappyFace.jpg -file
    HappyFace.jpg  -StoredCredentials UserDaveAccountA
   ```

Step 4: Clean up

1. After you're done testing, you can do the following to clean up:

   - Sign in to the AWS Management Console using Account A credentials, and do the following:

- In the Amazon S3 console, remove the bucket policy attached to *amzn-s3-demo-bucket1*. In the bucket **Properties**, delete the policy in the **Permissions** section.
- If the bucket is created for this exercise, in the Amazon S3 console, delete the objects and then delete the bucket.
- In the IAM Console, remove the **AccountAadmin** user. For step-by-step instructions, see Deleting an IAM user in the *IAM User Guide*.

2. Sign in to the AWS Management Console using Account B credentials. In the IAM Console, delete the user **AccountBadmin**.

Example 4 - Bucket owner granting cross-account permission to objects it does not own

Topics

- Understanding cross-account permissions and using IAM roles
- Step 0: Preparing for the walkthrough
- Step 1: Do the account A tasks
- Step 2: Do the Account B tasks
- Step 3: Do the Account C tasks
- Step 4: Clean up
- Related resources

In this example scenario, you own a bucket and you have enabled other AWS accounts to upload objects. If you have applied the bucket owner enforced setting for S3 Object Ownership for the bucket, you will own all objects in the bucket, including objects written by another AWS account. This approach resolves the issue that objects are not owned by you, the bucket owner. Then, you can delegate permission to users in your own account or to other AWS accounts. Suppose the bucket owner enforced setting for S3 Object Ownership is not enabled. That is, your bucket can have objects that other AWS accounts own.

Now, suppose as a bucket owner, you need to grant cross-account permission on objects, regardless of who the owner is, to a user in another account. For example, that user could be a billing application that needs to access object metadata. There are two core issues:

- The bucket owner has no permissions on those objects created by other AWS accounts. For the bucket owner to grant permissions on objects it doesn't own, the object owner must first grant permission to the bucket owner. The object owner is the AWS account that created the objects. The bucket owner can then delegate those permissions.
- The bucket owner account can delegate permissions to users in its own account (see Example 3: Bucket owner granting permissions to objects it does not own). However, the bucket owner account can't delegate permissions to other AWS accounts because cross-account delegation isn't supported.

In this scenario, the bucket owner can create an AWS Identity and Access Management (IAM) role with permission to access objects Then, the bucket owner can grant another AWS account permission to assume the role, temporarily enabling it to access objects in the bucket.

> **Note**
>
> S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.
> A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

Understanding cross-account permissions and using IAM roles

IAM roles enable several scenarios to delegate access to your resources, and cross-account access is one of the key scenarios. In this example, the bucket owner, Account A, uses an IAM role to temporarily delegate object access cross-account to users in another AWS account, Account C. Each IAM role that you create has the following two policies attached to it:

- A trust policy identifying another AWS account that can assume the role.

- An access policy defining what permissions—for example, `s3:GetObject`—are allowed when someone assumes the role. For a list of permissions you can specify in a policy, see Policy actions for Amazon S3.

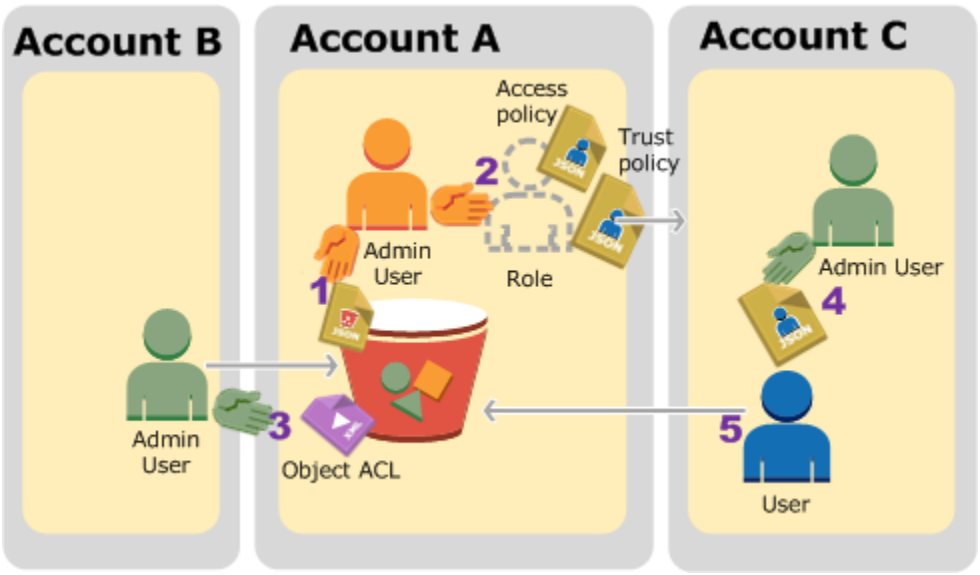
The AWS account identified in the trust policy then grants its user permission to assume the role. The user can then do the following to access objects:

- Assume the role and, in response, get temporary security credentials.
- Using the temporary security credentials, access the objects in the bucket.

For more information about IAM roles, see IAM Roles in the *IAM User Guide*.

1. Account A administrator user attaches a bucket policy granting Account B conditional permission to upload objects.
2. Account A administrator creates an IAM role, establishing trust with Account C, so users in that account can access Account A. The access policy attached to the role limits what user in Account C can do when the user accesses Account A.
3. Account B administrator uploads an object to the bucket owned by Account A, granting full-control permission to the bucket owner.
4. Account C administrator creates a user and attaches a user policy that allows the user to assume the role.
5. User in Account C first assumes the role, which returns the user temporary security credentials. Using those temporary credentials, the user then accesses objects in the bucket.

The following is a summary of the walkthrough steps:



For this example, you need three accounts. The following table shows how we refer to these accounts and the administrator users in these accounts. In accordance with the IAM guidelines (see About using an administrator user to create resources and grant permissions), we don't use the AWS account root user credentials in this walkthrough. Instead, you create an administrator user in each account and use those credentials when creating resources and granting them permissions.

AWS account ID	Account referred to as	Administrator user in the account
1111-1111-1111	Account A	AccountAadmin
2222-2222-2222	Account B	AccountBadmin
3333-3333-3333	Account C	AccountCadmin

Step 0: Preparing for the walkthrough

> **Note**
>
> You might want to open a text editor, and write down some of the information as you go through the steps. In particular, you will need account IDs, canonical user IDs, IAM user Sign-in URLs for each account to connect to the console, and Amazon Resource Names (ARNs) of the IAM users, and roles.

1. Make sure that you have three AWS accounts and each account has one administrator user as shown in the table in the preceding section.

   a. Sign up for AWS accounts, as needed. We refer to these accounts as Account A, Account B, and Account C.

   b. Using Account A credentials, sign in to the IAM console and do the following to create an administrator user:

      - Create user **`AccountAadmin`** and note its security credentials. For more information about adding users, see Creating an IAM user in your AWS account in the *IAM User Guide*.
      - Grant administrator privileges to **AccountAadmin** by attaching a user policy giving full access. For instructions, see Managing IAM policies in the *IAM User Guide*.
      - In the IAM Console **Dashboard**, note the **IAM User Sign-In URL**. Users in this account must use this URL when signing in to the AWS Management Console. For more information, see Sign in to the AWS Management Console as an IAM user in the *IAM User Guide*.

   c. Repeat the preceding step to create administrator users in Account B and Account C.

2. For Account C, note the canonical user ID.

   When you create an IAM role in Account A, the trust policy grants Account C permission to assume the role by specifying the account ID. You can find account information as follows:

   a. Use your AWS account ID or account alias, your IAM user name, and your password to sign in to the Amazon S3 console.

   b. Choose the name of an Amazon S3 bucket to view the details about that bucket.

   c. Choose the **Permissions** tab and then choose **Access Control List**.

   d. In the **Access for your AWS account** section, in the **Account** column is a long identifier, such as `c1daexampleaaf850ea79cf0430f33d72579fd1611c97f7ded193374c0b163b6`. This is your canonical user ID.

3. When creating a bucket policy, you will need the following information. Note these values:

   - **Canonical user ID of Account A** – When the Account A administrator grants conditional upload object permission to the Account B administrator, the condition specifies the canonical user ID of the Account A user that must get full-control of the objects.

> **Note**
>
> The canonical user ID is the Amazon S3–only concept. It is a 64-character obfuscated version of the account ID.

- **User ARN for Account B administrator** – You can find the user ARN in the IAM Console.You must select the user and find the user's ARN in the **Summary** tab.

  In the bucket policy, you grant `AccountBadmin` permission to upload objects and you specify the user using the ARN. Here's an example ARN value:

  ```
  arn:aws:iam::AccountB-ID:user/AccountBadmin
  ```

4. Set up either the AWS Command Line Interface (CLI) or the AWS Tools for Windows PowerShell. Make sure that you save administrator user credentials as follows:

   - If using the AWS CLI, create profiles, `AccountAadmin` and `AccountBadmin`, in the config file.
   - If using the AWS Tools for Windows PowerShell, make sure that you store credentials for the session as `AccountAadmin` and `AccountBadmin`.

   For instructions, see Setting up the tools for the walkthroughs.

Step 1: Do the account A tasks

In this example, Account A is the bucket owner. So user AccountAadmin in Account A will do the following:

- Create a bucket.
- Attach a bucket policy that grants the Account B administrator permission to upload objects.
- Create an IAM role that grants Account C permission to assume the role so it can access objects in the bucket.

Step 1.1: Sign in to the AWS Management Console

Using the IAM user Sign-in URL for Account A, first sign in to the AWS Management Console as **`AccountAadmin`** user. This user will create a bucket and attach a policy to it.

Step 1.2: Create a bucket and attach a bucket policy

In the Amazon S3 console, do the following:

1. Create a bucket. This exercise assumes the bucket name is *amzn-s3-demo-bucket1*.

   For instructions, see Creating a general purpose bucket.

2. Attach the following bucket policy. The policy grants conditional permission to the Account B administrator permission to upload objects.

   Update the policy by providing your own values for *amzn-s3-demo-bucket1*, *AccountB-ID*, and the *CanonicalUserId-of-AWSaccountA-BucketOwner*.

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "111",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        },
        {
            "Sid": "112",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=CanonicalUserId-of-
AWSaccountA-BucketOwner"
                }
            }
        }
```

```
    ]
}
```

Step 1.3: Create an IAM role to allow Account C cross-account access in Account A

In the IAM Console, create an IAM role (**examplerole**) that grants Account C permission to assume the role. Make sure that you are still signed in as the Account A administrator because the role must be created in Account A.

1. Before creating the role, prepare the managed policy that defines the permissions that the role requires. You attach this policy to the role in a later step.

   a. In the navigation pane on the left, choose **Policies** and then choose **Create Policy**.

   b. Next to **Create Your Own Policy**, choose **Select**.

   c. Enter **access-accountA-bucket** in the **Policy Name** field.

   d. Copy the following access policy and paste it into the **Policy Document** field. The access policy grants the role `s3:GetObject` permission so, when the Account C user assumes the role, it can only perform the `s3:GetObject` operation.

      JSON

      ```
      {
        "Version":"2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*"
          }
        ]
      }
      ```

   e. Choose **Create Policy**.

      The new policy appears in the list of managed policies.

2. In the navigation pane on the left, choose **Roles** and then choose **Create New Role**.

3. Under **Select Role Type**, select **Role for Cross-Account Access**, and then choose the **Select** button next to **Provide access between AWS accounts you own**.

4. Enter the Account C account ID.

   For this walkthrough, you don't need to require users to have multi-factor authentication (MFA) to assume the role, so leave that option unselected.

5. Choose **Next Step** to set the permissions that will be associated with the role.

6. Select the checkbox next to the **access-accountA-bucket** policy that you created, and then choose **Next Step**.

   The Review page appears so you can confirm the settings for the role before it's created. One very important item to note on this page is the link that you can send to your users who need to use this role. Users who use the link go straight to the **Switch Role** page with the Account ID and Role Name fields already filled in. You can also see this link later on the **Role Summary** page for any cross-account role.

7. Enter `examplerole` for the role name, and then choose **Next Step**.

8. After reviewing the role, choose **Create Role**.

   The `examplerole` role is displayed in the list of roles.

9. Choose the role name `examplerole`.

10. Select the **Trust Relationships** tab.

11. Choose **Show policy document** and verify the trust policy shown matches the following policy.

    The following trust policy establishes trust with Account C, by allowing it the `sts:AssumeRole` action. For more information, see AssumeRole in the *AWS Security Token Service API Reference*.

    JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
            "Action": "sts:AssumeRole"
```

```
        }
    ]
}
```

12. Note the Amazon Resource Name (ARN) of the `examplerole` role that you created.

    Later in the following steps, you attach a user policy to allow an IAM user to assume this role, and you identify the role by the ARN value.

Step 2: Do the Account B tasks

The example bucket owned by Account A needs objects owned by other accounts. In this step, the Account B administrator uploads an object using the command line tools.

- Using the `put-object` AWS CLI command, upload an object to *amzn-s3-demo-bucket1*.

  ```
  aws s3api put-object --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --
  body HappyFace.jpg --grant-full-control id="canonicalUserId-ofTheBucketOwner" --
  profile AccountBadmin
  ```

  Note the following:

  - The `--Profile` parameter specifies the `AccountBadmin` profile, so the object is owned by Account B.
  - The parameter `grant-full-control` grants the bucket owner full-control permission on the object as required by the bucket policy.
  - The `--body` parameter identifies the source file to upload. For example, if the file is on the C: drive of a Windows computer, you specify `c:\HappyFace.jpg`.

Step 3: Do the Account C tasks

In the preceding steps, Account A has already created a role, `examplerole`, establishing trust with Account C. This role allows users in Account C to access Account A. In this step, the Account C administrator creates a user (Dave) and delegates him the `sts:AssumeRole` permission it received from Account A. This approach allows Dave to assume the `examplerole` and temporarily gain access to Account A. The access policy that Account A attached to the role limits what Dave can do when he accesses Account A—specifically, get objects in *amzn-s3-demo-bucket1*.

Step 3.1: Create a user in Account C and delegate permission to assume examplerole

1. Using the IAM user sign-in URL for Account C, first sign in to the AWS Management Console as **`AccountCadmin`** user.

2. In the IAM Console, create a user, Dave.

   For step-by-step instructions, see Creating IAM users (AWS Management Console) in the *IAM User Guide*.

3. Note the Dave credentials. Dave will need these credentials to assume the `examplerole` role.

4. Create an inline policy for the Dave IAM user to delegate the `sts:AssumeRole` permission to Dave on the `examplerole` role in Account A.

   a. In the navigation pane on the left, choose **Users**.

   b. Choose the user name **Dave**.

   c. On the user details page, select the **Permissions** tab and then expand the **Inline Policies** section.

   d. Choose **click here** (or **Create User Policy**).

   e. Choose **Custom Policy**, and then choose **Select**.

   f. Enter a name for the policy in the **Policy Name** field.

   g. Copy the following policy into the **Policy Document** field.

      You must update the policy by providing the *`AccountA-ID`*.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/examplerole"
        }
    ]
```

```
}
```

h. Choose **Apply Policy**.

5. Save Dave's credentials to the config file of the AWS CLI by adding another profile, `AccountCDave`.

```
[profile AccountCDave]
aws_access_key_id = UserDaveAccessKeyID
aws_secret_access_key = UserDaveSecretAccessKey
region = us-west-2
```

Step 3.2: Assume role (examplerole) and access objects

Now Dave can access objects in the bucket owned by Account A as follows:

- Dave first assumes the `examplerole` using his own credentials. This will return temporary credentials.
- Using the temporary credentials, Dave will then access objects in Account A's bucket.

1. At the command prompt, run the following AWS CLI `assume-role` command using the `AccountCDave` profile.

   You must update the ARN value in the command by providing the *`AccountA-ID`* where `examplerole` is defined.

```
aws sts assume-role --role-arn arn:aws:iam::AccountA-ID:role/examplerole --profile
 AccountCDave --role-session-name test
```

   In response, AWS Security Token Service (AWS STS) returns temporary security credentials (access key ID, secret access key, and a session token).

2. Save the temporary security credentials in the AWS CLI config file under the `TempCred` profile.

```
[profile TempCred]
aws_access_key_id = temp-access-key-ID
aws_secret_access_key = temp-secret-access-key
aws_session_token = session-token
region = us-west-2
```

3. At the command prompt, run the following AWS CLI command to access objects using the temporary credentials. For example, the command specifies the head-object API to retrieve object metadata for the `HappyFace.jpg` object.

```
aws s3api get-object --bucket amzn-s3-demo-bucket1 --
key HappyFace.jpg SaveFileAs.jpg --profile TempCred
```

Because the access policy attached to `examplerole` allows the actions, Amazon S3 processes the request. You can try any other action on any other object in the bucket.

If you try any other action—for example, `get-object-acl`—you will get permission denied because the role isn't allowed that action.

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket1 --key HappyFace.jpg --
profile TempCred
```

We used user Dave to assume the role and access the object using temporary credentials. It could also be an application in Account C that accesses objects in *amzn-s3-demo-bucket1*. The application can obtain temporary security credentials, and Account C can delegate the application permission to assume `examplerole`.

Step 4: Clean up

1. After you're done testing, you can do the following to clean up:

   - Sign in to the AWS Management Console using Account A credentials, and do the following:

     - In the Amazon S3 console, remove the bucket policy attached to *amzn-s3-demo-bucket1*. In the bucket **Properties**, delete the policy in the **Permissions** section.
     - If the bucket is created for this exercise, in the Amazon S3 console, delete the objects and then delete the bucket.
     - In the IAM Console, remove the `examplerole` you created in Account A. For step-by-step instructions, see Deleting an IAM user in the *IAM User Guide*.
     - In the IAM Console, remove the **AccountAadmin** user.

2. Sign in to the IAM Console by using Account B credentials. Delete the user **AccountBadmin**.

3. Sign in to the IAM Console by using Account C credentials. Delete **AccountCadmin** and the user Dave.

Related resources

For more information that's related to this walkthrough, see the following resources in the *IAM User Guide*:

- Creating a role to delegate permissions to an IAM user
- Tutorial: Delegate Access Across AWS accounts Using IAM Roles
- Managing IAM policies

Using service-linked roles for Amazon S3 Storage Lens

To use Amazon S3 Storage Lens to collect and aggregate metrics across all your accounts in AWS Organizations, you must first ensure that S3 Storage Lens has trusted access enabled by the management account in your organization. S3 Storage Lens creates a service-linked role (SLR) to allow it to get the list of AWS accounts belonging to your organization. This list of accounts is used by S3 Storage Lens to collect metrics for S3 resources in all the member accounts when the S3 Storage Lens dashboard or configurations are created or updated.

Amazon S3 Storage Lens uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to S3 Storage Lens. Service-linked roles are predefined by S3 Storage Lens and include all the permissions that the service requires to call other AWS services on your behalf.

A service-linked role makes setting up S3 Storage Lens easier because you don't have to add the necessary permissions manually. S3 Storage Lens defines the permissions of its service-linked roles, and unless defined otherwise, only S3 Storage Lens can assume its roles. The defined permissions include the trust policy and the permissions policy, and that permissions policy can't be attached to any other IAM entity.

You can delete this service-linked role only after first deleting the related resources. This protects your S3 Storage Lens resources because you can't inadvertently remove permission to access the resources.

For information about other services that support service-linked roles, see AWS services that work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Service-linked role permissions for Amazon S3 Storage Lens

S3 Storage Lens uses the service-linked role named **AWSServiceRoleForS3StorageLens** – This enables access to AWS services and resources used or managed by S3 Storage Lens. This allows S3 Storage Lens to access AWS Organizations resources on your behalf.

The S3 Storage Lens service-linked role trusts the following service on your organization's storage:

- `storage-lens.s3.amazonaws.com`

The role permissions policy allows S3 Storage Lens to complete the following actions:

- `organizations:DescribeOrganization`

  `organizations:ListAccounts`

  `organizations:ListAWSServiceAccessForOrganization`

  `organizations:ListDelegatedAdministrators`

You must configure permissions to allow an IAM entity (such as a user, group, or role) to create, edit, or delete a service-linked role. For more information, see Service-linked role permissions in the *IAM User Guide*.

Creating a service-linked role for S3 Storage Lens

You don't need to manually create a service-linked role. When you complete one of the following tasks while signed into the AWS Organizations management or the delegate administrator accounts, S3 Storage Lens creates the service-linked role for you:

- Create an S3 Storage Lens dashboard configuration for your organization in the Amazon S3 console.
- PUT an S3 Storage Lens configuration for your organization using the REST API, AWS CLI and SDKs.

> **Note**
>
> S3 Storage Lens will support a maximum of five delegated administrators per organization.

If you delete this service-linked role, the preceding actions will re-create it as needed.

Example policy for S3 Storage Lens service-linked role

Example Permissions policy for the S3 Storage Lens service-linked role

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AwsOrgsAccess",
            "Effect": "Allow",
            "Action": [
                "organizations:DescribeOrganization",
                "organizations:ListAccounts",
                "organizations:ListAWSServiceAccessForOrganization",
                "organizations:ListDelegatedAdministrators"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Editing a service-linked role for Amazon S3 Storage Lens

S3 Storage Lens doesn't allow you to edit the AWSServiceRoleForS3StorageLens service-linked role. After you create a service-linked role, you can't change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM. For more information, see Editing a service-linked role in the *IAM User Guide*.

Deleting a service-linked role for Amazon S3 Storage Lens

If you no longer need to use the service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up the resources for your service-linked role before you can manually delete it.

> **Note**
>
> If the Amazon S3 Storage Lens service is using the role when you try to delete the resources, then the deletion might fail. If that happens, wait for a few minutes and try the operation again.

To delete the AWSServiceRoleForS3StorageLens you must delete all the organization level S3 Storage Lens configurations present in all AWS Regions using the AWS Organizations management or the delegate administrator accounts.

The resources are organization-level S3 Storage Lens configurations. Use S3 Storage Lens to clean up the resources and then use the IAM Console, CLI, REST API, or AWS SDK to delete the role.

In the REST API, AWS CLI, and SDKs, S3 Storage Lens configurations can be discovered using `ListStorageLensConfigurations` in all the Regions where your organization has created S3 Storage Lens configurations. Use the action `DeleteStorageLensConfiguration` to delete these configurations so that you can then delete the role.

> **Note**
>
> To delete the service-linked role, you must delete all the organization-level S3 Storage Lens configurations in all the Regions where they exist.

To delete Amazon S3 Storage Lens resources used by the AWSServiceRoleForS3StorageLens SLR

1. To get a list of your organization level configurations, you must use the `ListStorageLensConfigurations` in every Region that you have S3 Storage Lens configurations. This list can also be obtained from the Amazon S3 console.
2. Delete these configurations from the appropriate Regional endpoints by invoking the `DeleteStorageLensConfiguration` API call or by using the Amazon S3 console.

To manually delete the service-linked role using IAM

After you have deleted the configurations, delete the AWSServiceRoleForS3StorageLens SLR from the IAM Console or by invoking the IAM API `DeleteServiceLinkedRole`, or using the AWS CLI or AWS SDK. For more information, see Deleting a service-linked role in the *IAM User Guide*.

Supported Regions for S3 Storage Lens service-linked roles

S3 Storage Lens supports using service-linked roles in all of the AWS Regions where the service is available. For more information, see Amazon S3 Regions and Endpoints.

Troubleshooting Amazon S3 identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with Amazon S3 and IAM.

Topics

- I received an access denied error
- I am not authorized to perform an action in Amazon S3
- I am not authorized to perform iam:PassRole
- I want to allow people outside of my AWS account to access my Amazon S3 resources
- Troubleshoot access denied (403 Forbidden) errors in Amazon S3

I received an access denied error

Verify that there is not an explicit `Deny` statement against the requester you are trying to grant permissions to in either the bucket policy or the identity-based policy.

For detailed information about troubleshooting access denied errors, see Troubleshoot access denied (403 Forbidden) errors in Amazon S3.

I am not authorized to perform an action in Amazon S3

If you receive an error that you're not authorized to perform an action, your policies must be updated to allow you to perform the action.

The following example error occurs when the `mateojackson` IAM user tries to use the console to view details about a fictional *`my-example-widget`* resource but doesn't have the fictional `s3:GetWidget` permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 s3:GetWidget on resource: my-example-widget
```

In this case, the policy for the `mateojackson` user must be updated to allow access to the *`my-example-widget`* resource by using the `s3:GetWidget` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to Amazon S3.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in Amazon S3. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my Amazon S3 resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether Amazon S3 supports these features, see How Amazon S3 works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see Cross account resource access in IAM in the *IAM User Guide*.

Troubleshoot access denied (403 Forbidden) errors in Amazon S3

Access denied (HTTP `403 Forbidden`) errors appear when AWS explicitly or implicitly denies an authorization request.

- An *explicit denial* occurs when a policy contains a `Deny` statement for the specific AWS action.
- An *implicit denial* occurs when there is no applicable `Deny` statement and also no applicable `Allow` statement.

Because an AWS Identity and Access Management (IAM) policy implicitly denies an IAM principal by default, the policy must explicitly allow the principal to perform an action. Otherwise, the policy implicitly denies access. For more information, see The difference between explicit and implicit denies in the *IAM User Guide*. For information about the policy evaluation logic that determines whether an access request is allowed or denied, see Policy evaluation logic in the *IAM User Guide*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

The following topics cover the most common causes of access denied errors in Amazon S3.

> **Note**
>
> For access denied (HTTP `403 Forbidden`) errors, Amazon S3 doesn't charge the bucket owner when the request is initiated outside of the bucket owner's individual AWS account or the bucket owner's AWS organization.

Topics

- Access denied message examples and how to troubleshoot them
- Access denied due to Requester Pays settings
- Bucket policies and IAM policies
- Amazon S3 ACL settings
- S3 Block Public Access settings
- Amazon S3 encryption settings
- S3 Object Lock settings
- VPC endpoint policies
- AWS Organizations policies
- CloudFront distribution access
- Access point settings
- Additional resources

> **Note**
>
> If you're trying to troubleshoot a permissions issue, start with the the section called "Access denied message examples and how to troubleshoot them" section, then go to the ??? section. Also be sure to follow the guidance in the section called "Tips for checking permissions".

Access denied message examples and how to troubleshoot them

Amazon S3 now includes additional context in access denied (HTTP `403 Forbidden`) errors for requests made to resources within the same AWS account or same organization in AWS Organizations. This context includes the type of policy that denied access, the reason for denial, and information about the IAM user or role that requested access to the resource.

With this additional context, you can troubleshoot access issues, identify the root cause of access denied errors, and fix incorrect access controls by updating the relevant policies. This additional context is also available in AWS CloudTrail logs. Enhanced access denied error messages for same-account or same-organization requests are available in all AWS Regions, including the AWS GovCloud (US) Regions and the China Regions.

For explicit deny cases, the error message also includes the Amazon Resource Name (ARN) of the specific policy that denied the request. With this policy ARN, you can quickly identify the exact policy responsible for the denial and navigate directly to it to make the necessary changes. The error message includes the policy ARN for Service Control Policies (SCPs), resource control policies (RCPs), identity-based policies, session policies, and permissions boundaries.

Most access denied error messages appear in the format `User user-arn is not authorized to perform action on "resource-arn" because context`. In this example, *`user-arn`* is the Amazon Resource Name (ARN) of the user that doesn't receive access, *`action`* is the service action that the policy denies, and *`resource-arn`* is the ARN of the resource on which the policy acts. The *`context`* field represents additional context about the policy type that explains why the policy denied access.

When a policy explicitly denies access because the policy contains a `Deny` statement, then the access denied error message includes the phrase `with an explicit deny in a type policy`, along with the ARN of the specific policy that denied the request. When the policy implicitly denies access, then the access denied error message includes the phrase `because no type policy allows the action action`.

Important

- Enhanced access denied messages are returned only for same-account requests or for requests within the same organization in AWS Organizations. Cross-account requests outside of the same organization return a generic `Access Denied` message.

  For information about the policy evaluation logic that determines whether a cross-account access request is allowed or denied, see Cross-account policy evaluation logic in the *IAM User Guide*. For a walkthrough that shows how to grant cross-account access, see the section called "Granting cross-account permissions".
- For requests within the same organization in AWS Organizations:
  - Enhanced access denied messages aren't returned if a denial occurs because of a virtual private cloud (VPC) endpoint policy.
  - Enhanced access denied messages are provided whenever both the bucket owner and the caller account belong to the same organization in AWS Organizations. Although buckets configured with the S3 Object Ownership **Bucket owner preferred** or **Object writer** settings might contain objects owned by different accounts, object ownership doesn't affect enhanced access denied messages. Enhanced access denied messages

are returned for all object requests as long as the bucket owner and caller are in the same organization, regardless of who owns the specific object. For information about Object Ownership settings and configurations, see the section called "Controlling object ownership".

- Enhanced access denied error messages aren't returned for requests made to directory buckets. Directory bucket requests return a generic `Access Denied` message.
- If multiple policies of the same policy type deny an authorization request, the access denied error message doesn't specify the number of policies.
- For explicit deny cases, the error message includes the ARN of the specific policy that denied the request. This applies to Service Control Policies (SCPs), resource control policies (RCPs), identity-based policies, session policies, and permissions boundaries.
- If multiple policy types deny an authorization request, the error message includes only one of those policy types.
- If an access request is denied due to multiple reasons, the error message includes only one of the reasons for denial.

The following examples show the format for different types of access denied error messages and how to troubleshoot each type of message.

Access denied due to Blocked Encryption Type

To limit the server-side encryption types you can use in your general purpose buckets, you can choose to block SSE-C write requests by updating your default encryption configuration for your buckets. This bucket-level configuration blocks requests to upload objects that specify SSE-C. When SSE-C is blocked for a bucket, any `PutObject`, `CopyObject`, `PostObject`, or Multipart Upload or replication requests that specify SSE-C encryption will be rejected with an HTTP 403 `AccessDenied` error.

This setting is a parameter on the `PutBucketEncryption` API and can also be updated using the S3 Console, AWS CLI, and AWS SDKs, if you have the `s3:PutEncryptionConfiguration` permission. Valid values are `SSE-C`, which blocks SSE-C encryption for the general purpose bucket, and `NONE`, which allows the use SSE-C for writes to the bucket.

For example, when access is denied for a `PutObject` request because the `BlockedEncryptionTypes` setting blocks write requests specifying SSE-C, you receive the following message:

```
An error occurred (AccessDenied) when calling the PutObject operation:
User: arn:aws:iam::123456789012:user/MaryMajor  is not
authorized to perform: s3:PutObject on resource:
"arn:aws:s3:::amzn-s3-demo-bucket1/object-name" because this
bucket has blocked upload requests that specify
Server Side Encryption with Customer provided keys (SSE-C).
Please specify a different server-side encryption type
```

For more information about this setting, see the section called “Blocking or unblocking SSE-C”.

Access denied due to a resource control policy – explicit denial

1. Check for an explicit `Deny` statement for the action in your resource control policies (RCPs). The error message includes the policy ARN, which you can use to identify and navigate directly to the policy that denied the request. For the following example, the action is `s3:GetObject`.
2. Update your RCP by removing the `Deny` statement. For more information, see Update a resource control policy (RCP) in the *AWS Organizations User Guide*.

```
An error occurred (AccessDenied) when calling the GetObject operation:
User: arn:aws:iam::777788889999:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name"
with an explicit deny in a resource control policy, with policy ARN:
arn:aws:organizations::777788889999:policy/o-exampleorgid/resource_control_policy/p-
examplepolicyid
```

Access denied due to a Service Control Policy – implicit denial

1. Check for a missing `Allow` statement for the action in your service control policies (SCPs). For the following example, the action is `s3:GetObject`.
2. Update your SCP by adding the `Allow` statement. For more information, see Updating an SCP in the *AWS Organizations User Guide*.

```
User: arn:aws:iam::777788889999:user/MaryMajor is not authorized to perform:
s3:GetObject because no service control policy allows the s3:GetObject action
```

Access denied due to a Service Control Policy – explicit denial

1. Check for an explicit `Deny` statement for the action in your Service Control Policies (SCPs). The error message includes the policy ARN, which you can use to identify and navigate directly to the policy that denied the request. For the following example, the action is `s3:GetObject`.
2. Update your SCP by changing the `Deny` statement to allow the user the necessary access. For an example of how you can do this, see Service control policy examples on GitHub. For more information about updating your SCP, see Updating an SCP in the *AWS Organizations User Guide*.

```
User: arn:aws:iam::777788889999:user/MaryMajor is not authorized to perform:
s3:GetObject with an explicit deny in a service control policy, with policy ARN:
arn:aws:organizations::777788889999:policy/o-exampleorgid/service_control_policy/p-
examplepolicyid
```

Access denied due to a VPC endpoint policy – implicit denial

1. Check for a missing `Allow` statement for the action in your virtual private cloud (VPC) endpoint policies. For the following example, the action is `s3:GetObject`.
2. Update your VPC endpoint policy by adding the `Allow` statement. For more information, see Update a VPC endpoint policy in the *AWS PrivateLink Guide*.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no VPC endpoint policy allows the s3:GetObject action
```

Access denied due to a VPC endpoint policy – explicit denial

1. Check for an explicit `Deny` statement for the action in your virtual private cloud (VPC) endpoint policies. For the following example, the action is `s3:GetObject`.
2. Update your VPC endpoint policy by changing the `Deny` statement to allow the user the necessary access. For example, you can update your `Deny` statement to use the `aws:PrincipalAccount` condition key with the `StringNotEquals` condition operator to allow the specific principal access, as shown in the section called "Example 7: Excluding principals from Deny statements". For more information about updating your VPC endpoint policy, see Update a VPC endpoint policy in the *AWS PrivateLink Guide*.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
```

```
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in a VPC endpoint policy
```

Access denied due to a permissions boundary – implicit denial

1. Check for a missing `Allow` statement for the action in your permissions boundary. For the following example, the action is `s3:GetObject`.
2. Update your permissions boundary by adding the `Allow` statement to your IAM policy. For more information, see Permissions boundaries for IAM entities and Editing IAM policies in the *IAM User Guide*.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name"
because no permissions boundary allows the s3:GetObject action
```

Access denied due to a permissions boundary – explicit denial

1. Check for an explicit `Deny` statement for the action in your permissions boundary. The error message includes the policy ARN, which you can use to identify and navigate directly to the policy that denied the request. For the following example, the action is `s3:GetObject`.
2. Update your permissions boundary by changing the `Deny` statement in your IAM policy to allow the user the necessary access. For example, you can update your `Deny` statement to use the `aws:PrincipalAccount` condition key with the `StringNotEquals` condition operator to allow the specific principal access, as shown in aws:PrincipalAccount in the *IAM User Guide*. For more information, see Permissions boundaries for IAM entities and Editing IAM policies in the *IAM User Guide*.

```
User: arn:aws:iam::777788889999:user/MaryMajor is not authorized to perform:
s3:GetObject with an explicit deny in a permissions boundary, with policy ARN:
arn:aws:iam::777788889999:policy/ExamplePermissionsBoundaryPolicy
```

Access denied due to session policies – implicit denial

1. Check for a missing `Allow` statement for the action in your session policies. For the following example, the action is `s3:GetObject`.
2. Update your session policy by adding the `Allow` statement. For more information, see Session policies and Editing IAM policies in the *IAM User Guide*.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no session policy allows the s3:GetObject action
```

Access denied due to session policies – explicit denial

1. Check for an explicit `Deny` statement for the action in your session policies. The error message includes the policy ARN, which you can use to identify and navigate directly to the policy that denied the request. For the following example, the action is `s3:GetObject`.
2. Update your session policy by changing the `Deny` statement to allow the user the necessary access. For example, you can update your `Deny` statement to use the `aws:PrincipalAccount` condition key with the `StringNotEquals` condition operator to allow the specific principal access, as shown in the section called "Example 7: Excluding principals from Deny statements". For more information about updating your session policy, see Session policies and Editing IAM policies in the *IAM User Guide*.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in a session policy, with policy ARN:
arn:aws:iam::123456789012:policy/ExampleSessionPolicy
```

Access denied due to resource-based policies – implicit denial

> **Note**
>
> *Resource-based policies* means policies such as bucket policies and access point policies.

1. Check for a missing `Allow` statement for the action in your resource-based policy. Also check whether the `IgnorePublicAcls` S3 Block Public Access setting is applied on the bucket, access point, or account level. For the following example, the action is `s3:GetObject`.
2. Update your policy by adding the `Allow` statement. For more information, see Resource-based policies and Editing IAM policies in the *IAM User Guide*.

   You might also need to adjust your `IgnorePublicAcls` block public access setting for the bucket, access point, or account. For more information, see the section called "Access denied due to Block Public Access settings" and the section called "Configuring bucket and access point settings".

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no resource-based policy allows the s3:GetObject action
```

Access denied due to resource-based policies – explicit denial

> **Note**
>
> *Resource-based policies* means policies such as bucket policies and access point policies.

1. Check for an explicit `Deny` statement for the action in your resource-based policy. Also check whether the `RestrictPublicBuckets` S3 Block Public Access setting is applied on the bucket, access point, or account level. For the following example, the action is `s3:GetObject`.
2. Update your policy by changing the `Deny` statement to allow the user the necessary access. For example, you can update your `Deny` statement to use the `aws:PrincipalAccount` condition key with the `StringNotEquals` condition operator to allow the specific principal access, as shown in the section called "Example 7: Excluding principals from Deny statements". For more information about updating your resource-based policy, see Resource-based policies and Editing IAM policies in the *IAM User Guide*.

   You might also need to adjust your `RestrictPublicBuckets` block public access setting for the bucket, access point, or account. For more information, see the section called "Access denied due to Block Public Access settings" and the section called "Configuring bucket and access point settings".

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in a resource-based policy
```

Access denied due to identity-based policies – implicit denial

1. Check for a missing `Allow` statement for the action in identity-based policies attached to the identity. For the following example, the action is `s3:GetObject` and the identity is the IAM user `MaryMajor`.
2. Update your policy by adding the `Allow` statement. For more information, see Identity-based policies and Editing IAM policies in the *IAM User Guide*.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject because no identity-based policy allows the s3:GetObject action
```

Access denied due to identity-based policies – explicit denial

1. Check for an explicit `Deny` statement for the action in identity-based policies attached to the identity. The error message includes the policy ARN, which you can use to identify and navigate directly to the policy that denied the request. For the following example, the action is `s3:GetObject` and the identity is the IAM user `MaryMajor`.
2. Update your policy by changing the `Deny` statement to allow the user the necessary access. For example, you can update your `Deny` statement to use the `aws:PrincipalAccount` condition key with the `StringNotEquals` condition operator to allow the specific principal access, as shown in aws:PrincipalAccount in the *IAM User Guide*. For more information, see Identity-based policies and Editing IAM policies in the *IAM User Guide*.

```
User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
an explicit deny in an identity-based policy, with policy ARN:
arn:aws:iam::123456789012:policy/ExampleIdentityBasedPolicy
```

Access denied due to Block Public Access settings

The Amazon S3 Block Public Access feature provides settings for access points, buckets, and accounts to help you manage public access to Amazon S3 resources. For more information about how Amazon S3 defines "public," see The meaning of "public".

By default, new buckets, access points, and objects don't allow public access. However, users can modify bucket policies, access point policies, IAM user policies, object permissions, or access control lists (ACLs) to allow public access. S3 Block Public Access settings override these policies, permissions, and ACLs. Since April 2023, all Block Public Access settings are enabled by default for new buckets.

When Amazon S3 receives a request to access a bucket or an object, it determines whether the bucket or the bucket owner's account has a block public access setting applied. If the request was made through an access point, Amazon S3 also checks for block public access settings for the access point. If there is an existing block public access setting that prohibits the requested access, Amazon S3 rejects the request.

Amazon S3 Block Public Access provides four settings. These settings are independent and can be used in any combination. Each setting can be applied to an access point, a bucket, or an entire AWS account. If the block public access settings for the access point, bucket, or account differ, then Amazon S3 applies the most restrictive combination of the access point, bucket, and account settings.

When Amazon S3 evaluates whether an operation is prohibited by a block public access setting, it rejects any request that violates an access point, bucket, or account setting.

The four settings provided by Amazon S3 Block Public Access are as follows:

- `BlockPublicAcls` – This setting applies to `PutBucketAcl`, `PutObjectAcl`, `PutObject`, `CreateBucket`, `CopyObject`, and `POST Object` requests. The `BlockPublicAcls` setting causes the following behavior:
  - `PutBucketAcl` and `PutObjectAcl` calls fail if the specified access control list (ACL) is public.
  - `PutObject` calls fail if the request includes a public ACL.
  - If this setting is applied to an account, then `CreateBucket` calls fail with an HTTP `400` (`Bad Request`) response if the request includes a public ACL.

  For example, when access is denied for a `CopyObject` request because of the `BlockPublicAcls` setting, you receive the following message:

  ```
  An error occurred (AccessDenied) when calling the CopyObject operation:
  User: arn:aws:sts::123456789012:user/MaryMajor is not authorized to
  perform: s3:CopyObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name"
  because public ACLs are prevented by the BlockPublicAcls setting in S3 Block Public
   Access.
  ```

- `IgnorePublicAcls` – The `IgnorePublicAcls` setting causes Amazon S3 to ignore all public ACLs on a bucket and any objects that it contains. If your request's permission is granted only by a public ACL, then the `IgnorePublicAcls` setting rejects the request.

  Any denial resulting from the `IgnorePublicAcls` setting is implicit. For example, if `IgnorePublicAcls` denies a `GetObject` request because of a public ACL, you receive the following message:

  ```
  User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
  s3:GetObject because no resource-based policy allows the s3:GetObject action
  ```

- `BlockPublicPolicy` – This setting applies to `PutBucketPolicy` and `PutAccessPointPolicy` requests.

  Setting `BlockPublicPolicy` for a bucket causes Amazon S3 to reject calls to `PutBucketPolicy` if the specified bucket policy allows public access. This setting also causes Amazon S3 to reject calls to `PutAccessPointPolicy` for all of the bucket's same-account access points if the specified policy allows public access.

  Setting `BlockPublicPolicy` for an access point causes Amazon S3 to reject calls to `PutAccessPointPolicy` and `PutBucketPolicy` that are made through the access point if the specified policy (for either the access point or the underlying bucket) allows public access.

  For example, when access is denied on a `PutBucketPolicy` request because of the `BlockPublicPolicy` setting, you receive the following message:

  ```
  An error occurred (AccessDenied) when calling the PutBucketPolicy operation:
  User: arn:aws:sts::123456789012:user/MaryMajor is not authorized to
  perform: s3:PutBucketPolicy on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-
  name"
  because public policies are prevented by the BlockPublicPolicy setting in S3 Block
   Public Access.
  ```

- `RestrictPublicBuckets` – The `RestrictPublicBuckets` setting restricts access to an access point or bucket with a public policy to only AWS service principals and authorized users within the bucket owner's account and the access point owner's account. This setting blocks all cross-account access to the access point or bucket (except by AWS service principals), while still allowing users within the account to manage the access point or bucket. This setting also rejects all anonymous (or unsigned) calls.

  Any denial resulting from the `RestrictPublicBuckets` setting is explicit. For example, if `RestrictPublicBuckets` denies a `GetObject` request because of a public bucket or access point policy, you receive the following message:

  ```
  User: arn:aws:iam::123456789012:user/MaryMajor is not authorized to perform:
  s3:GetObject on resource: "arn:aws:s3:::amzn-s3-demo-bucket1/object-name" with
  an explicit deny in a resource-based policy
  ```

For more information about these settings, see the section called “Block public access settings”. To review and update these settings, see the section called “Configuring block public access”.

Access denied due to Requester Pays settings

If the Amazon S3 bucket you are trying to access has the Requester Pays feature enabled, you need to make sure you are passing the correct request parameters when making requests to that bucket. The Requester Pays feature in Amazon S3 allows the requester, instead of the bucket owner, to pay the data transfer and request costs for accessing objects in the bucket. When Requester Pays is enabled for a bucket, the bucket owner is not charged for requests made by other AWS accounts.

If you make a request to a Requester Pays-enabled bucket without passing the necessary parameters, you will receive an Access Denied (403 Forbidden) error. To access objects in a Requester Pays-enabled bucket, you must do the following:

1. When making requests using the AWS CLI, you must include the `--request-payer requester` parameter. For example, to copy an object with the key `object.txt` located in the `s3://amzn-s3-demo-bucket/` S3 bucket to a location on your local machine, you must also pass the parameter `--request-payer requester` if this bucket has Requester Pays enabled.

   ```
   aws s3 cp s3://amzn-s3-demo-bucket/object.txt /local/path \
   --request-payer requester
   ```

2. When making programmatic requests using an AWS SDK, set the `x-amz-request-payer` header to the value `requester`. For an example, see Downloading objects from Requester Pays buckets.
3. Make sure that the IAM user or role making the request has the necessary permissions to access the Requester Pays bucket, such as the `s3:GetObject` and `s3:ListBucket` permissions.

By including the `--request-payer requester` parameter or setting the `x-amz-request-payer` header, you are informing Amazon S3 that you, the requester, will pay the costs associated with accessing the objects in the Requester Pays-enabled bucket. This will prevent the Access Denied (403 Forbidden) error.

Bucket policies and IAM policies

Bucket-level operations

If there is no bucket policy in place, then the bucket implicitly allows requests from any AWS Identity and Access Management (IAM) identity in the bucket-owner's account. The bucket also implicitly denies requests from any other IAM identities from any other accounts, and anonymous (unsigned) requests. However, if there is no IAM user policy in place, the requester (unless they're

the AWS account root user) is implicitly denied from making any requests. For more information about this evaluation logic, see Determining whether a request is denied or allowed within an account in the *IAM User Guide*.

Object-level operations

If the object is owned by the bucket-owning account, the bucket policy and IAM user policy will function in the same way for object-level operations as they do for bucket-level operations. For example, if there is no bucket policy in place, then the bucket implicitly allows object requests from any IAM identity in the bucket-owner's account. The bucket also implicitly denies object requests from any other IAM identities from any other accounts, and anonymous (unsigned) requests. However, if there is no IAM user policy in place, the requester (unless they're the AWS account root user) is implicitly denied from making any object requests.

If the object is owned by an external account, then access to the object can be granted only through object access control lists (ACLs). The bucket policy and IAM user policy can still be used to deny object requests.

Therefore, to ensure that your bucket policy or IAM user policy isn't causing an Access Denied (403 Forbidden) error, make sure that the following requirements are met:

- For same-account access, there must not be an explicit `Deny` statement against the requester you are trying to grant permissions to, in either the bucket policy or the IAM user policy. If you want to grant permissions by using only the bucket policy and the IAM user policy, there must be at least one explicit `Allow` statement in one of these policies.
- For cross-account access, there must not be an explicit `Deny` statement against the requester that you're trying to grant permissions to, in either the bucket policy or the IAM user policy. To grant cross-account permissions by using only the bucket policy and IAM user policy, make sure that both the bucket policy and the IAM user policy of the requester include an explicit `Allow` statement.

> **Note**
>
> `Allow` statements in a bucket policy apply only to objects that are owned by the same bucket-owning account. However, `Deny` statements in a bucket policy apply to all objects regardless of object ownership.

To review or edit your bucket policy

> **Note**
>
> To view a bucket policy, you must have the `s3:GetBucketPolicy` permission. To edit a bucket policy, you must have the `s3:PutBucketPolicy` permission.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. From the **Buckets** list, choose the name of the bucket that you want to view or edit a bucket policy for.
4. Choose the **Permissions** tab.
5. Under **Bucket policy**, choose **Edit**. The **Edit bucket policy** page appears.

To review your bucket policy by using the AWS Command Line Interface (AWS CLI), use the get-bucket-policy command. To edit your bucket policy, use the put-bucket-policy command.

> **Note**
>
> If you get locked out of a bucket because of an incorrect bucket policy, sign in to the AWS Management Console by using your AWS account root user credentials. To regain access to your bucket, make sure to delete the incorrect bucket policy by using your AWS account root user credentials.

Tips for checking permissions

To check whether the requester has proper permissions to perform an Amazon S3 operation, try the following:

- Identify the requester. If it's an unsigned request, then it's an anonymous request without an IAM user policy. If it's a request that uses a presigned URL, then the user policy is the same as the one for the IAM user or role that signed the request.
- Verify that you're using the correct IAM user or role. You can verify your IAM user or role by checking the console navigation bar or by using the aws sts get-caller-identity command.

- Check the IAM policies that are related to the IAM user or role. You can use one of the following methods:
  - Test IAM policies with the IAM policy simulator.
  - Review the different IAM policy types.
- If needed, edit your IAM user policy.
- Review the following examples of policies that explicitly deny or allow access:
  - Explicit allow IAM user policy: IAM: Allows and denies access to multiple services programmatically and in the console
  - Explicit allow bucket policy: Granting permissions to multiple accounts to upload objects or set object ACLs for public access
  - Explicit deny IAM user policy: AWS: Denies access to AWS based on the requested AWS Region
  - Explicit deny bucket policy: Require SSE-KMS for all objects written to a bucket

Amazon S3 ACL settings

When checking your ACL settings, first review your Object Ownership setting to check whether ACLs are enabled on the bucket. Be aware that ACL permissions can be used only to grant permissions and can't be used to reject requests. ACLs also can't be used to grant access to requesters that are rejected by explicit denials in bucket policies or IAM user policies.

The Object Ownership setting is set to Bucket owner enforced

If the **Bucket owner enforced** setting is enabled, then ACL settings are unlikely to cause an Access Denied (403 Forbidden) error because this setting disables all ACLs that apply to bucket and objects. **Bucket owner enforced** is the default (and recommended) setting for Amazon S3 buckets.

The Object Ownership setting is set to Bucket owner preferred or Object writer

ACL permissions are still valid with the **Bucket owner preferred** setting or the **Object writer** setting. There are two kinds of ACLs: bucket ACLs and object ACLs. For the differences between these two types of ACLs, see Mapping of ACL permissions and access policy permissions.

Depending on the action of the rejected request, check the ACL permissions for your bucket or the object:

- If Amazon S3 rejected a `LIST`, PUT object, `GetBucketAcl`, or `PutBucketAcl` request, then review the ACL permissions for your bucket.

> **Note**
>
> You can't grant GET object permissions with bucket ACL settings.

- If Amazon S3 rejected a GET request on an S3 object, or a PutObjectAcl request, then review the ACL permissions for the object.

> **Important**
>
> If the account that owns the object is different from the account that owns the bucket, then access to the object isn't controlled by the bucket policy.

Troubleshooting an Access Denied (403 Forbidden) error from a GET object request during cross-account object ownership

Review the bucket's Object Ownership settings to determine the object owner. If you have access to the object ACLs, then you can also check the object owner's account. (To view the object owner's account, review the object ACL setting in the Amazon S3 console.) Alternatively, you can also make a `GetObjectAcl` request to find the object owner's canonical ID to verify the object owner account. By default, ACLs grant explicit allow permissions for GET requests to the object owner's account.

After you've confirmed that the object owner is different from the bucket owner, then depending on your use case and access level, choose one of the following methods to help address the Access Denied (403 Forbidden) error:

- **Disable ACLs (recommended)** – This method will apply to all objects and can be performed by the bucket owner. This method automatically gives the bucket owner ownership and full control over every object in the bucket. Before you implement this method, check the prerequisites for disabling ACLs. For information about how to set your bucket to **Bucket owner enforced** (recommended) mode, see Setting Object Ownership on an existing bucket.

> **Important**
>
> To prevent an Access Denied (403 Forbidden) error, be sure to migrate the ACL permissions to a bucket policy before you disable ACLs. For more information, see Bucket policy examples for migrating from ACL permissions.

- **Change the object owner to the bucket owner** – This method can be applied to individual objects, but only the object owner (or a user with the appropriate permissions) can change an object's ownership. Additional PUT costs might apply. (For more information, see Amazon S3 pricing.) This method grants the bucket owner full ownership of the object, allowing the bucket owner to control access to the object through a bucket policy.

  To change the object's ownership, do one of the following:

  - You (the bucket owner) can copy the object back to the bucket.
  - You can change the Object Ownership setting of the bucket to **Bucket owner preferred**. If versioning is disabled, the objects in the bucket are overwritten. If versioning is enabled, duplicate versions of the same object will appear in the bucket, which the bucket owner can set a lifecycle rule to expire. For instructions on how to change your Object Ownership setting, see Setting Object Ownership on an existing bucket.

    > **Note**
    >
    > When you update your Object Ownership setting to **Bucket owner preferred**, the setting is applied only to new objects that are uploaded to the bucket.

  - You can have the object owner upload the object again with the `bucket-owner-full-control` canned object ACL.

  > **Note**
  >
  > For cross-account uploads, you can also require the `bucket-owner-full-control` canned object ACL in your bucket policy. For an example bucket policy, see Grant cross-account permissions to upload objects while ensuring that the bucket owner has full control.

- **Keep the object writer as the object owner** – This method doesn't change the object owner, but it does allow you to grant access to objects individually. To grant access to an object, you

must have the `PutObjectAcl` permission for the object. Then, to fix the Access Denied (403 Forbidden) error, add the requester as a grantee to access the object in the object's ACLs. For more information, see Configuring ACLs.

S3 Block Public Access settings

If the failed request involves public access or public policies, then check the S3 Block Public Access settings on your account, bucket, or access point. For more information about troubleshooting access denied errors related to S3 Block Public Access settings, see the section called "Access denied due to Block Public Access settings".

Amazon S3 encryption settings

Amazon S3 supports server-side encryption on your bucket. Server-side encryption is the encryption of data at its destination by the application or service that receives it. Amazon S3 encrypts your data at the object level as it writes it to disks in AWS data centers and decrypts it for you when you access it.

By default, Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Amazon S3 also allows you to specify the server-side encryption method when uploading objects.

To review your bucket's server-side encryption status and encryption settings

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. From the **Buckets** list, choose the bucket that you want to check the encryption settings for.
4. Choose the **Properties** tab.
5. Scroll down to the **Default encryption** section and view the **Encryption type** settings.

To check your encryption settings by using the AWS CLI, use the get-bucket-encryption command.

To check the encryption status of an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.

3. From the **Buckets** list, choose the name of the bucket that contains the object.
4. From the **Objects** list, choose the name of the object that you want to add or change encryption for.

   The object's details page appears.
5. Scroll down to the **Server-side encryption settings** section to view the object's server-side encryption settings.

To check your object encryption status by using the AWS CLI, use the head-object command.

Encryption and permissions requirements

Amazon S3 supports three types of server-side encryption:

- Server-side encryption with Amazon S3 managed keys (SSE-S3)
- Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)
- Server-side encryption with customer-provided keys (SSE-C)

Based on your encryption settings, make sure that the following permissions requirements are met:

- **SSE-S3** – No extra permissions are required.
- **SSE-KMS (with a customer managed key)** – To upload objects, the `kms:GenerateDataKey` permission on the AWS KMS key is required. To download objects and perform multipart uploads of objects, the `kms:Decrypt` permission on the KMS key is required.
- **SSE-KMS (with an AWS managed key)** – The requester must be from the same account that owns the `aws/s3` KMS key. The requester must also have the correct Amazon S3 permissions to access the object.
- **SSE-C (with a customer provided key)** – No additional permissions are required. You can configure the bucket policy to require and restrict server-side encryption with customer-provided encryption keys for objects in your bucket.

If the object is encrypted with a customer managed key, make sure that the KMS key policy allows you to perform the `kms:GenerateDataKey` or `kms:Decrypt` actions. For instructions on checking your KMS key policy, see Viewing a key policy in the *AWS Key Management Service Developer Guide*.

S3 Object Lock settings

If your bucket has S3 Object Lock enabled and the object is protected by a retention period or legal hold and you try to delete an object, Amazon S3 returns one of the following responses, depending on how you tried to delete the object:

- **Permanent DELETE request** – If you issued a permanent `DELETE` request (a request that specifies a version ID), Amazon S3 returns an Access Denied (`403 Forbidden`) error when you try to delete the object.
- **Simple DELETE request** – If you issued a simple `DELETE` request (a request that doesn't specify a version ID), Amazon S3 returns a `200 OK` response and inserts a delete marker in the bucket, and that marker becomes the current version of the object with a new ID.

To check whether the bucket has Object Lock enabled

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. From the **Buckets** list, choose the name of the bucket that you want to review.
4. Choose the **Properties** tab.
5. Scroll down to the **Object Lock** section. Verify whether the **Object Lock** setting is **Enabled** or **Disabled**.

To determine whether the object is protected by a retention period or legal hold, view the lock information for your object.

If the object is protected by a retention period or legal hold, check the following:

- If the object version is protected by the compliance retention mode, there is no way to permanently delete it. A permanent `DELETE` request from any requester, including the AWS account root user, will result in an Access Denied (403 Forbidden) error. Also, be aware that when you submit a `DELETE` request for an object that's protected by the compliance retention mode, Amazon S3 creates a delete marker for the object.
- If the object version is protected with governance retention mode and you have the `s3:BypassGovernanceRetention` permission, you can bypass the protection and permanently delete the version. For more information, see Bypassing governance mode.

- If the object version is protected by a legal hold, then a permanent `DELETE` request can result in an Access Denied (403 Forbidden) error. To permanently delete the object version, you must remove the legal hold on the object version. To remove a legal hold, you must have the `s3:PutObjectLegalHold` permission. For more information about removing a legal hold, see Configuring S3 Object Lock.

VPC endpoint policies

If you're accessing Amazon S3 by using a virtual private cloud (VPC) endpoint, make sure that the VPC endpoint policy isn't blocking you from accessing your Amazon S3 resources. By default, the VPC endpoint policy allows all requests to Amazon S3. You can also configure the VPC endpoint policy to restrict certain requests. For information about how to check your VPC endpoint policy, see the following resources:

- the section called “Access denied due to a VPC endpoint policy – implicit denial”
- the section called “Access denied due to a VPC endpoint policy – explicit denial”
- Control access to VPC endpoints by using endpoint policies in the *AWS PrivateLink Guide*

AWS Organizations policies

If your AWS account belongs to an organization, AWS Organizations policies can block you from accessing Amazon S3 resources. By default, AWS Organizations policies don't block any requests to Amazon S3. However, make sure that your AWS Organizations policies haven't been configured to block access to S3 buckets. For instructions on how to check your AWS Organizations policies, see the following resources:

- the section called “Access denied due to a Service Control Policy – implicit denial”
- the section called “Access denied due to a Service Control Policy – explicit denial”
- the section called “Access denied due to a resource control policy – explicit denial”
- Listing all policies in the *AWS Organizations User Guide*

Additionally, if you incorrectly configured your bucket policy for a member account to deny all users access to your S3 bucket, you can unlock the bucket by launching a privileged session for the member account in IAM. After you launch a privileged session, you can delete the misconfigured bucket policy to regain access to the bucket. For more information, see Perform a privileged task on an AWS Organizations member account in the *AWS Identity and Access Management User Guide*.

CloudFront distribution access

If you receive a Access Denied (403 Forbidden) error when trying to access your S3 static website through CloudFront, check these common issues:

- **Do you have the correct origin domain name format?**
  - Make sure you're using the S3 website endpoint format (bucket-name.s3-website-region.amazonaws.com) rather than the REST API endpoint
  - Verify that static website hosting is enabled on your bucket
- **Does your bucket policy allow CloudFront access?**
  - Ensure your bucket policy includes permissions for your CloudFront distribution's Origin Access Identity (OAI) or Origin Access Control (OAC)
  - Verify the policy includes the required s3:GetObject permissions

For additional troubleshooting steps and configurations, including error page setups and protocol settings, see Why do I get a "403 access denied" error when I use an Amazon S3 website endpoint as the origin of my CloudFront distribution? in the AWS re:Post Knowledge Center.

> **Note**
>
> This error is different from 403 errors you might receive when accessing S3 directly. For CloudFront-specific issues, make sure to check both your CloudFront distribution settings and S3 configurations.

Access point settings

If you receive an Access Denied (403 Forbidden) error while making requests through Amazon S3 access points, you might need to check the following:

- The configurations for your access points
- The IAM user policy that's used for your access points
- The bucket policy that's used to manage or configure your cross-account access points

Access point configurations and policies

- When you create an access point, you can choose to designate **Internet** or **VPC** as the network origin. If the network origin is set to VPC only, Amazon S3 will reject any requests made to the access point that don't originate from the specified VPC. To check the network origin of your access point, see Creating access points restricted to a virtual private cloud.
- With access points, you can also configure custom Block Public Access settings, which work similarly to the Block Public Access settings at the bucket or account level. To check your custom Block Public Access settings, see Managing public access to access points for general purpose buckets.
- To make successful requests to Amazon S3 by using access points, make sure that the requester has the necessary IAM permissions. For more information, see Configuring IAM policies for using access points.
- If the request involves cross-account access points, make sure that the bucket owner has updated the bucket policy to authorize requests from the access point. For more information, see Granting permissions for cross-account access points.

If the Access Denied (403 Forbidden) error still persists after checking all the items in this topic, retrieve your Amazon S3 request ID and contact Support for additional guidance.

Additional resources

For more guidance on Access Denied (403 Forbidden) errors you can check the following resources:

- How do I troubleshoot 403 Access Denied errors from Amazon S3? in the AWS re:Post Knowledge Center.
- Why am I getting a "403 Forbidden" error when I try to upload files in Amazon S3? in the AWS re:Post Knowledge Center.
- Why do I get an Access Denied error when I try to access an Amazon S3 resource in the same AWS account? in the AWS re:Post Knowledge Center.
- Why do I get an Access Denied error when I try to access an Amazon S3 bucket with public read access? in the AWS re:Post Knowledge Center.
- Why do I get a "signature mismatch" error when I try to use a presigned URL to upload an object to Amazon S3? in the AWS re:Post Knowledge Center.
- Why do I get an Access Denied error for ListObjectsV2 when I run the sync command on my Amazon S3 bucket? in the AWS re:Post Knowledge Center.

- Why do I get a "403 access denied" error when I use an Amazon S3 website endpoint as the origin of my CloudFront distribution? in the AWS re:Post Knowledge Center.

AWS managed policies for Amazon S3

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: AmazonS3FullAccess

You can attach the `AmazonS3FullAccess` policy to your IAM identities. This policy grants permissions that allow full access to Amazon S3.

To view the permissions for this policy, see AmazonS3FullAccess in the AWS Management Console.

AWS managed policy: AmazonS3ReadOnlyAccess

You can attach the `AmazonS3ReadOnlyAccess` policy to your IAM identities. This policy grants permissions that allow read-only access to Amazon S3.

To view the permissions for this policy, see AmazonS3ReadOnlyAccess in the AWS Management Console.

AWS managed policy: AmazonS3ObjectLambdaExecutionRolePolicy

Provides AWS Lambda functions the required permissions to send data to S3 Object Lambda when requests are made to an S3 Object Lambda access point. Also grants Lambda permissions to write to Amazon CloudWatch logs.

To view the permissions for this policy, see AmazonS3ObjectLambdaExecutionRolePolicy in the AWS Management Console.

AWS managed policy: S3UnlockBucketPolicy

If you incorrectly configured your bucket policy for a member account to deny all users access to your S3 bucket, you can use this AWS managed policy (`S3UnlockBucketPolicy`) to unlock the bucket. For more information on how to remove a misconfigured bucket policy that denies all principals from accessing an Amazon S3 bucket, see Perform a privileged task on an AWS Organizations member account in the *AWS Identity and Access Management User Guide*.

Amazon S3 updates to AWS managed policies

View details about updates to AWS managed policies for Amazon S3 since this service began tracking these changes.

Change	Description	Date
Amazon S3 added `S3UnlockBucketPolicy`	Amazon S3 added a new AWS-managed policy called `S3UnlockBucketPolicy` to unlock a bucket and remove a misconfigured bucket policy that denies all principals from accessing an Amazon S3 bucket.	November 1, 2024
Amazon S3 added Describe permissions to `AmazonS3ReadOnlyAccess`	Amazon S3 added `s3:Describe*` permissions to `AmazonS3ReadOnlyAccess` .	August 11, 2023

Change	Description	Date
Amazon S3 added S3 Object Lambda permissions to `AmazonS3FullAccess` and `AmazonS3ReadOnlyAccess`	Amazon S3 updated the `AmazonS3FullAccess` and `AmazonS3ReadOnlyAccess` policies to include permissions for S3 Object Lambda.	September 27, 2021
Amazon S3 added `AmazonS3ObjectLambdaExecutionRolePolicy`	Amazon S3 added a new AWS-managed policy called `AmazonS3ObjectLambdaExecutionRolePolicy` that provides Lambda functions permissions to interact with S3 Object Lambda and write to CloudWatch logs.	August 18, 2021
Amazon S3 started tracking changes	Amazon S3 started tracking changes for its AWS managed policies.	August 18, 2021

Managing access to shared datasets with access points

Amazon S3 access points simplify data access for any AWS service or customer application that stores data in S3. Access points are named network endpoints that are attached to a data source such as a bucket, Amazon FSx for NetApp ONTAP volume, Amazon FSx for OpenZFS volume, or an Amazon S3 recovery point in AWS Backup. For information about working with buckets, see General purpose buckets overview. For information about working with FSx for NetApp ONTAP, see What is Amazon FSx for NetApp ONTAP in the *FSx for ONTAP User Guide*. For information about working with FSx for OpenZFS, see What is Amazon FSx for OpenZFS in the *FSx for OpenZFS User Guide*. An Amazon S3 recovery point (or backup) represents the content of an S3 bucket at a specified point in time, created by AWS Backup. For information about working with Amazon S3 backups, see Amazon S3 backups in the *AWS Backup Developer Guide*.

You can use access points to perform S3 object operations, such as `GetObject` and `PutObject`. Each access point has distinct permissions and network controls that S3 applies for any request that is made through that access point. Each endpoint enforces a customized access point policy that allow you to control use by resource, user, or other conditions. If your access point is attached to a bucket the access point policy works in conjunction with the underlying bucket policy. You can configure any access point to accept requests only from a virtual private cloud (VPC) to restrict Amazon S3 data access to a private network. You can also configure custom block public access settings for each access point.

> **Note**
>
> You can only use access points to perform operations on objects. You can't use access points to perform other Amazon S3 operations, such as deleting buckets or creating S3 Replication configurations. For a complete list of S3 operations that support access points, see Access point compatibility.

The topics in this section explain how to work with Amazon S3 access points. For topics on using access points with directory buckets see, Managing access to shared datasets in directory buckets with access points.

Topics

- Access points naming rules, restrictions, and limitations
- Referencing access points with ARNs, access point aliases, or virtual-hosted–style URIs
- Access point compatibility
- Configuring IAM policies for using access points
- Monitoring and logging access points
- Creating an access point
- Managing your Amazon S3 access points for general purpose buckets
- Using Amazon S3 access points for general purpose buckets
- Using tags with S3 Access Points for general purpose buckets

Access points naming rules, restrictions, and limitations

Access points are named network endpoints attached to a bucket, a volume on an Amazon FSx file system, or an Amazon S3 recovery point in AWS Backup that simplify managing data. When you create an access point you choose a name and the AWS Region to create it in. The following topics provide information about access point naming rules, restrictions and limitations.

Topics

- Naming rules for access points
- Restrictions and limitations for access points
- Restrictions and limitations for access points attached to a volume on an Amazon FSx file system
- Restrictions and limitations for access points attached to an Amazon S3 recovery point in AWS Backup

Naming rules for access points

When you create an access point, you choose its name and the AWS Region to create it in. Unlike general purpose buckets access point names do not need to be unique across AWS accounts or AWS Regions. The same AWS account may create access points with the same name in different AWS Regions or two different AWS accounts may use the same access point name. However, within a single AWS Region an AWS account may not have two identically named access points.

> **Note**
>
> If you choose to publicize your access point name, avoid including sensitive information in the access point name. Access point names are published in a publicly accessible database known as the Domain Name System (DNS).

Access point names must be DNS-compliant and must meet the following conditions:

- Must be unique within a single AWS account and AWS Region
- Must begin with a number or lowercase letter
- Must be between 3 and 50 characters long
- Can't begin or end with a hyphen (-)
- Can't contain underscores (_), uppercase letters, spaces, or periods (.)

- Can't end with the suffix `-s3alias` or `-ext-s3alias`. These suffixes are reserved for access point alias names. For more information, see Access point aliases.

Restrictions and limitations for access points

Amazon S3 access points have the following restrictions and limitations:

- Each access point is associated with exactly one bucket or FSx for OpenZFS volume. You must specify this when you create the access point. After you create an access point, you can't associate it with a different bucket or FSx for OpenZFS volume. However, you can delete an access point, and then create another one with the same name.
- After you create an access point, you can't change its virtual private cloud (VPC) configuration.
- Access point policies are limited to 20 KB in size.
- You can create a maximum of 10,000 access points per AWS account per AWS Region. If you need more than 10,000 access points for a single account in a single Region, you can request a service quota increase. For more information about service quotas and requesting an increase, see AWS service quotas in the *AWS General Reference*.
- You can't use an access point as a destination for S3 Replication. For more information about replication, see Replicating objects within and across Regions.
- You can't use S3 access point aliases as the source or destination for **Move** operations in the Amazon S3 console.
- You can address access points only by using virtual-host-style URLs. For more information about virtual-host-style addressing, see Accessing an Amazon S3 general purpose bucket.
- API operations that control access point functionality (for example, `PutAccessPoint` and `GetAccessPointPolicy`) don't support cross-account calls.
- You must use AWS Signature Version 4 when making requests to an access point by using the REST APIs. For more information about authenticating requests, see Authenticating Requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.
- Access points only support requests over HTTPS. Amazon S3 will automatically respond with an HTTP redirect for any requests made over HTTP, to upgrade the request to HTTPS.
- Access points don't support anonymous access.
- After you create an access point, you can't change its block public access settings.
- Cross-account access points don’t grant you access to data until you are granted permissions from the bucket owner. The bucket owner always retains ultimate control over access to the data

and must update the bucket policy to authorize requests from the cross-account access point. To view a bucket policy example, see Configuring IAM policies for using access points.

- In AWS Regions where you have more than 1,000 access points, you can't search for an access point by name in the Amazon S3 console.
- When you're viewing a cross-account access point in the Amazon S3 console, the **Access** column displays **Unknown**. The Amazon S3 console can't determine if public access is granted for the associated bucket and objects. Unless you require a public configuration for a specific use case, we recommend that you and the bucket owner block all public access to the access point and the bucket. For more information, see Blocking public access to your Amazon S3 storage.

Restrictions and limitations for access points attached to a volume on an Amazon FSx file system

The following are specific limitations when using access points attached to a volume on an Amazon FSx file system:

- When creating an access points you can only attach the access point to a volume on a Amazon FSx file systems that you own. You cannot attach to a volume owned by another AWS account.
- You cannot use the `CreateAccessPoint` API when creating and attaching an access point to a volume on a Amazon FSx file system. You must use the CreateAndAttachS3AccessPoint API.
- You can not turn off any block public access settings when creating or using an access point attached to a volume on an Amazon FSx file system.
- You can't list objects or use **Copy** or **Move** operations in the S3 console with access points attached to a volume on an Amazon FSx file system.
- `CopyObject` is supported for access points attached to an FSx for NetApp ONTAP or FSx for OpenZFS volume only if the source and destination are the same access point. For more information, about access point compatibility, see Access point compatibility.
- Multipart uploads are limited to 5GB.
- FSx for OpenZFS deployment type and storage class support varies by AWS Region. For more information, see Availability by AWS Region in the *OpenZFS User Guide*.

Restrictions and limitations for access points attached to an Amazon S3 recovery point in AWS Backup

The following are specific limitations when using access points attached to an S3 recovery point in AWS Backup:

- Access points for S3 recovery points are read-only. Write operations such as `PutObject`, `DeleteObject`, `CopyObject` and multipart uploads are not supported.
- Use the AWS Backup `CreateBackupAccessPoint` API or the AWS Backup console to create access points. The S3 `CreateAccessPoint` API is not supported.
- To delete these access points use the AWS Backup `DeleteBackupAccessPoint` API or the AWS Backup console.
- Each account can have up to 5 access points per recovery point, regardless of status.
- Access point names share the S3 access point namespace. A name used by a backup access point cannot be reused for a standard S3 access point in the same account and Region.
- For more information, see Backup access points in the *AWS Backup Developer Guide*.

Referencing access points with ARNs, access point aliases, or virtual-hosted–style URIs

After you create an access point you can use these endpoints to preform a number of operations. When referring to an access point you can use the Amazon Resource Names (ARNs), access point alias, or virtual-hosted–style URI.

Topics

- Access point ARNs
- Access point aliases
- Virtual-hosted–style URI

Access point ARNs

Access points have Amazon Resource Names (ARNs). Access point ARNs are similar to bucket ARNs, but they are explicitly typed and encode the access point's AWS Region and the AWS account ID of the access point's owner. For more information about ARNs, see Identify AWS resources with Amazon Resource Names (ARNs) in the *IAM User Guide*.

Access point ARNs use the following format:

```
arn:aws:s3:region:account-id:accesspoint/resource
```

- `arn:aws:s3:us-west-2:123456789012:accesspoint/test` represents the access point named *test*, owned by account *123456789012* in the Region *us-west-2*.
- `arn:aws:s3:us-west-2:123456789012:accesspoint/*` represents all access points under account *123456789012* in the Region *us-west-2*.

ARNs for objects accessed through an access point use the following format:

```
arn:aws:s3:region:account-id:accesspoint/access-point-name/object/resource
```

- `arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01` represents the object *unit-01*, accessed through the access point named *test*, owned by account *123456789012* in the Region *us-west-2*.
- `arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/*` represents all objects for the access point named *test*, in account *123456789012* in the Region *us-west-2*.
- `arn:aws:s3:us-west-2:123456789012:accesspoint/test/object/unit-01/finance/*` represents all objects under prefix *unit-01/finance/* for the access point named *test*, in account *123456789012* in the Region *us-west-2*.

Access point aliases

When you create an access point, Amazon S3 automatically generates an alias that you can use instead of an Amazon S3 bucket name for data access. You can use this access point alias instead of an Amazon Resource Name (ARN) for access point data plane operations. For a list of these operations, see Access point compatibility.

An access point alias name is created within the same namespace as an Amazon S3 bucket. This alias name is automatically generated and cannot be changed. An access point alias name meets all the requirements of a valid Amazon S3 bucket name and consists of the following parts:

ACCESS POINT NAME`-METADATA-s3alias` (for access points attached to an Amazon S3 bucket)

ACCESS POINT NAME`-METADATA-ext-s3alias` (for access points attached to an non-S3 bucket data source)

> **Note**
>
> The `-s3alias` and `-ext-s3alias` suffixes are reserved for access point alias names and can't be used for bucket or access point names. For more information about Amazon S3 bucket-naming rules, see General purpose bucket naming rules.

Access points aliases use cases and limitations

When adopting access points, you can use access point alias names without requiring extensive code changes.

When you create an access point, Amazon S3 automatically generates an access point alias name, as shown in the following example. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-access-point --bucket amzn-s3-demo-bucket1 --name my-access-point
 --account-id 111122223333
{
    "AccessPointArn": "arn:aws:s3:region:111122223333:accesspoint/my-access-point",
    "Alias": "my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-s3alias"
}
```

You can use this access point alias name instead of an Amazon S3 bucket name in any data plane operation. For a list of these operations, see Access point compatibility.

The following AWS CLI example for the `get-object` command uses the bucket's access point alias to return information about the specified object. To run this command, replace the *`user input placeholders`* with your own information.

```
aws s3api get-object --bucket my-access-point-aqfqprnstn7aefdfbarligizwgyfouse1a-
s3alias --key dir/my_data.rtf my_data.rtf

{
    "AcceptRanges": "bytes",
    "LastModified": "2020-01-08T22:16:28+00:00",
    "ContentLength": 910,
    "ETag": "\"00751974dc146b76404bb7290f8f51bb\"",
    "VersionId": "null",
    "ContentType": "text/rtf",
    "Metadata": {}
```

```
}
```

Access point alias limitations

- Aliases cannot be configured by customers.
- Aliases cannot be deleted or modified or disabled on an access point.
- You can use this access point alias name instead of an Amazon S3 bucket name in some data plane operations. For a list of these operations, see Access points compatibility with S3 operations.
- You can't use an access point alias name for Amazon S3 control plane operations. For a list of Amazon S3 control plane operations, see Amazon S3 Control in the *Amazon Simple Storage Service API Reference*.
- You can't use S3 access point aliases as the source or destination for **Move** operations in the Amazon S3 console.
- Aliases cannot be used in AWS Identity and Access Management (IAM) policies.
- Aliases cannot be used as a logging destination for S3 server access logs.
- Aliases cannot be used as a logging destination for AWS CloudTrail logs.
- Amazon SageMaker AI GroundTruth does not support access point aliases.

Virtual-hosted–style URI

Access points only support virtual-host-style addressing. In a virtual-hosted–style URI, the access point name, AWS account, and AWS Region is part of the domain name in the URL. For more information about virtual hosting, see Virtual hosting of general purpose buckets.

Virtual-hosted–style URI for access points use the following format:

```
https://access-point-name-account-id.s3-accesspoint.region.amazonaws.com
```

Note

- If your access point name includes dash (-) characters, include the dashes in the URL and insert another dash before the account ID. For example, to use an access point named *finance-docs* owned by account *123456789012* in the Region *us-west-2*, the appropriate URL would be `https://finance-docs-123456789012.s3-accesspoint.us-west-2.amazonaws.com`.

- S3 access points don't support access through HTTP. Access points support only secure access through HTTPS.

Access point compatibility

You can use access points to access objects using the following subset of Amazon S3 APIs. All the operations listed below can accept either access point ARNs or access point aliases.

For examples of using access points to perform operations on objects, see Using Amazon S3 access points for general purpose buckets.

Access points compatibility with S3 operations

The following table is a partial list of Amazon S3 operations and if they are compatible with access points. All operations below are supported by access points using an S3 bucket as its data source, while only some operations are supported by access points using an FSx for ONTAP or FSx for OpenZFS volume or an S3 recovery point in AWS Backup as a data source.

For more information see, access point compatibility in the *FSx for ONTAP User Guide* or the *FSx for OpenZFS User Guide*.

S3 operation	Access point attached to an S3 bucket	Access point attached to an FSx for OpenZFS volume	Access point attached to an Amazon S3 recovery point in AWS Backup
AbortMultipartUpload	Supported	Supported	Not supported
CompleteMultipartUpload	Supported	Supported	Not supported
CopyObject (same-Region copies only)	Supported	Supported, if source and destination are the same access point	Not supported
CreateMultipartUpload	Supported	Supported	Not supported

S3 operation	Access point attached to an S3 bucket	Access point attached to an FSx for OpenZFS volume	Access point attached to an Amazon S3 recovery point in AWS Backup
DeleteObject	Supported	Supported	Not supported
DeleteObjects	Supported	Supported	Not supported
DeleteObjectTagging	Supported	Supported	Not supported
GetBucketAcl	Supported	Not supported	Not supported
GetBucketCors	Supported	Not supported	Not supported
GetBucketLocation	Supported	Supported	Supported
GetBucketNotificationConfiguration	Supported	Not supported	Not supported
GetBucketPolicy	Supported	Not supported	Not supported
GetObject	Supported	Supported	Supported
GetObjectAcl	Supported	Not supported	Not supported
GetObjectAttributes	Supported	Supported	Supported
GetObjectLegalHold	Supported	Not supported	Not supported
GetObjectRetention	Supported	Not supported	Not supported

S3 operation	Access point attached to an S3 bucket	Access point attached to an FSx for OpenZFS volume	Access point attached to an Amazon S3 recovery point in AWS Backup
GetObjectTagging	Supported	Supported	Supported
HeadBucket	Supported	Supported	Not supported
HeadObject	Supported	Supported	Supported
ListMultipartUploads	Supported	Supported	Not supported
ListObjects	Supported	Supported	Supported
ListObjectsV2	Supported	Supported	Supported
ListObjectVersions	Supported	Not supported	Supported
ListParts	Supported	Supported	Not supported
Presign	Supported	Supported	Not supported
PutObject	Supported	Supported	Not supported
PutObjectAcl	Supported	Not supported	Not supported
PutObjectLegalHold	Supported	Not supported	Not supported
PutObjectRetention	Supported	Not supported	Not supported
PutObjectTagging	Supported	Supported	Not supported
RestoreObject	Supported	Not supported	Not supported

S3 operation	Access point attached to an S3 bucket	Access point attached to an FSx for OpenZFS volume	Access point attached to an Amazon S3 recovery point in AWS Backup
UploadPart	Supported	Supported	Not supported
UploadPartCopy (same-Region copies only)	Supported	Supported, if source and destination are the same access point	Not supported

Configuring IAM policies for using access points

Amazon S3 access points support AWS Identity and Access Management (IAM) resource policies that allow you to control the use of the access point by resource, user, or other conditions. For an application or user to be able to access objects through an access point, both the access point and the underlying bucket or Amazon FSx file system must permit the request.

> **Important**
>
> Restrictions that you include in an access point policy apply only to requests made through that access point. Attaching an access point to a bucket does not change underlying resource's behavior. All existing operations against the bucket not made through your access point will continue to work as before.

When you're using IAM resource policies, make sure to resolve security warnings, errors, general warnings, and suggestions from AWS Identity and Access Management Access Analyzer before you save your policy. IAM Access Analyzer runs policy checks to validate your policy against IAM policy grammar and best practices. These checks generate findings and provide recommendations to help you author policies that are functional and conform to security best practices.

To learn more about validating policies by using IAM Access Analyzer, see IAM Access Analyzer policy validation in the *IAM User Guide*. To view a list of the warnings, errors, and suggestions that are returned by IAM Access Analyzer, see IAM Access Analyzer policy check reference.

Policy examples for access points

The following examples demonstrate how to create IAM policies to control requests made through an access point.

> **Note**
>
> Permissions granted in an access point policy are effective only if the underlying bucket also allows the same access. You can accomplish this in two ways:
>
> 1. **(Recommended)** Delegate access control from the bucket to the access point, as described in Delegating access control to access points.
> 2. Add the same permissions contained in the access point policy to the underlying bucket's policy. The Example 1 access point policy example demonstrates how to modify the underlying bucket policy to allow the necessary access.

Example 1 – Access point policy grant

The following access point policy grants IAM user *Jane* in account *123456789012* permissions to GET and PUT objects with the prefix *Jane/* through the access point *my-access-point* in account *123456789012*.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Jane"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-point/object/Jane/*"
    }]
}
```

Note

For the access point policy to effectively grant access to *Jane*, the underlying bucket must also allow the same access to *Jane*. You can delegate access control from the bucket to the access point as described in Delegating access control to access points. Or, you can add the following policy to the underlying bucket to grant the necessary permissions to Jane. Note that the `Resource` entry differs between the access point and bucket policies.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Jane"
        },
        "Action": ["s3:GetObject", "s3:PutObject"],
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/Jane/*"
    }]
}
```

Example 2 – Access point policy with tag condition

The following access point policy grants IAM user *Mateo* in account *123456789012* permissions to GET objects through the access point *my-access-point* in the account *123456789012* that have the tag key *data* set with a value of *finance*.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Mateo"
```

```
        },
        "Action": "s3:GetObject",
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-
point/object/*",
        "Condition": {
            "StringEquals": {
                "s3:ExistingObjectTag/data": "finance"
            }
        }
    }]
}
```

Example 3 – Access point policy that allows bucket listing

The following access point policy allows IAM user `Arnav` in the account *123456789012* permission to view the objects contained in the bucket underlying the access point *my-access-point* in the account *123456789012*.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::123456789012:user/Arnav"
        },
        "Action": "s3:ListBucket",
        "Resource": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-access-
point"
    }]
}
```

Example 4 – Service control policy

The following service control policy requires all new access points to be created with a virtual private cloud (VPC) network origin. With this policy in place, users in your organization can't create new access points that are accessible from the internet.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Effect": "Deny",
        "Action": "s3:CreateAccessPoint",
        "Resource": "*",
        "Condition": {
            "StringNotEquals": {
                "s3:AccessPointNetworkOrigin": "VPC"
            }
        }
    }]
}
```

Example 5 – Bucket policy to limit S3 operations to VPC network origins

The following bucket policy limits access to all S3 object operations for the bucket *amzn-s3-demo-bucket* to access points with a VPC network origin.

> **Important**
>
> Before using a statement like the one shown in this example, make sure that you don't need to use features that aren't supported by access points, such as Cross-Region Replication.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:AbortMultipartUpload",
```

```
                "s3:BypassGovernanceRetention",
                "s3:DeleteObject",
                "s3:DeleteObjectTagging",
                "s3:DeleteObjectVersion",
                "s3:DeleteObjectVersionTagging",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:GetObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:GetObjectTagging",
                "s3:GetObjectVersion",
                "s3:GetObjectVersionAcl",
                "s3:GetObjectVersionTagging",
                "s3:ListMultipartUploadParts",
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:PutObjectLegalHold",
                "s3:PutObjectRetention",
                "s3:PutObjectTagging",
                "s3:PutObjectVersionAcl",
                "s3:PutObjectVersionTagging",
                "s3:RestoreObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:AccessPointNetworkOrigin": "VPC"
                }
            }
        }
    ]
}
```

Condition keys

S3 access points have condition keys that you can use in IAM policies to control access to your resources. The following condition keys represent only part of an IAM policy. For full policy examples, see Policy examples for access points, the section called “Delegating access control to access points”, and the section called “Granting permissions for cross-account access points”.

`s3:DataAccessPointArn`

This example shows a string that you can use to match on an access point ARN. The following example matches all access points for AWS account *123456789012* in Region *us-west-2*:

```
"Condition" : {
    "StringLike": {
        "s3:DataAccessPointArn": "arn:aws:s3:us-west-2:123456789012:accesspoint/*"
    }
}
```

`s3:DataAccessPointAccount`

This example shows a string operator that you can use to match on the account ID of the owner of an access point. The following example matches all access points that are owned by the AWS account *123456789012*.

```
"Condition" : {
    "StringEquals": {
        "s3:DataAccessPointAccount": "123456789012"
    }
}
```

`s3:AccessPointNetworkOrigin`

This example shows a string operator that you can use to match on the network origin, either `Internet` or VPC. The following example matches only access points with a VPC origin.

```
"Condition" : {
    "StringEquals": {
        "s3:AccessPointNetworkOrigin": "VPC"
    }
}
```

For more information about using condition keys with Amazon S3, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Delegating access control to access points

You can delegate access control for a bucket to the bucket's access points. The following example bucket policy allows full access to all access points that are owned by the bucket owner's account. Thus, all access to this bucket is controlled by the policies attached to its access points. We recommend configuring your buckets this way for all use cases that don't require direct access to the bucket.

Example 6 – Bucket policy that delegates access control to access points

JSON

```
{
    "Version":"2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "arn:aws:s3:::amzn-s3-demo-bucket", "arn:aws:s3:::amzn-s3-
demo-bucket/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "111122223333" }
        }
    }]
}
```

Granting permissions for cross-account access points

To create an access point to a bucket that's owned by another account, you must first create the access point by specifying the bucket name and account owner ID. Then, the bucket owner must update the bucket policy to authorize requests from the access point. Creating an access point is similar to creating a DNS CNAME in that the access point doesn't provide access to the bucket contents. All bucket access is controlled by the bucket policy. The following example bucket policy allows `GET` and `LIST` requests on the bucket from an access point that's owned by a trusted AWS account.

Replace *`Bucket ARN`* with the ARN of the bucket.

Example 7 – Bucket policy delegating permissions to another AWS account

JSON

```
{
    "Version":"2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : ["s3:GetObject","s3:ListBucket"],
        "Resource" : ["arn:aws:s3:::amzn-s3-demo-bucket", "arn:aws:s3:::amzn-s3-
demo-bucket/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointAccount" : "Access point
 owner's account ID" }
        }
    }]
}
```

Note

Cross-account access points are only available for access points attached to S3 buckets. You cannot attach an access point to a volume on an Amazon FSx file system owned by another AWS account.

Monitoring and logging access points

Amazon S3 logs requests made through access points and requests made to the API operations that manage access points, such as `CreateAccessPoint` and `GetAccessPointPolicy`. To monitor and manage usage patterns, you can also configure Amazon CloudWatch Logs request metrics for access points.

Topics

- CloudWatch request metrics
- AWS CloudTrail logs for requests made through access points

CloudWatch request metrics

To understand and improve the performance of applications that are using access points, you can use CloudWatch for Amazon S3 request metrics. Request metrics help you monitor Amazon S3 requests to quickly identify and act on operational issues.

By default, request metrics are available at the bucket level. However, you can define a filter for request metrics using a shared prefix, object tags, or an access point. When you create an access point filter, the request metrics configuration includes requests to the access point that you specify. You can receive metrics, set alarms, and access dashboards to view real-time operations performed through this access point.

You must opt in to request metrics by configuring them in the console or by using the Amazon S3 API. Request metrics are available at 1-minute intervals after some latency for processing. Request metrics are billed at the same rate as CloudWatch custom metrics. For more information, see Amazon CloudWatch pricing.

To create a request metrics configuration that filters by access point, see Creating a metrics configuration that filters by prefix, object tag, or access point.

AWS CloudTrail logs for requests made through access points

You can log requests made through access points and requests made to the APIs that manage access points, such as `CreateAccessPoint` and `GetAccessPointPolicy`, by using server access logging and AWS CloudTrail.

CloudTrail log entries for requests made through access points include the access point ARN in the `resources` section of the log.

For example, suppose you have the following configuration:

- A bucket named *amzn-s3-demo-bucket1* in Region *us-west-2* that contains an object named *my-image.jpg*
- An access point named *my-bucket-ap* that is associated with *amzn-s3-demo-bucket1*
- An AWS account ID of *123456789012*

The following example shows the `resources` section of a CloudTrail log entry for the preceding configuration:

```
"resources": [
        {"type": "AWS::S3::Object",
            "ARN": "arn:aws:s3:::amzn-s3-demo-bucket1/my-image.jpg"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::Bucket",
            "ARN": "arn:aws:s3:::amzn-s3-demo-bucket1"
        },
        {"accountId": "123456789012",
            "type": "AWS::S3::AccessPoint",
            "ARN": "arn:aws:s3:us-west-2:123456789012:accesspoint/my-bucket-ap"
        }
    ]
```

If you are using an access point attached to a volume on an Amazon FSx file system, the `resources` section of a CloudTrail log entry will look different. For example:

```
"resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::FSx::Volume",
            "ARN": "arn:aws:fsx:us-east-1:123456789012:volume/fs-0123456789abcdef9/
fsvol-01234567891112223"
        }
    ]
```

If you are using an access point attached to an S3 recovery point in AWS Backup, data plane requests (such as `GetObject` and `ListObjectsV2`) are logged in CloudTrail the same way as requests through standard S3 access points. For example:

```
"resources": [
        {
            "accountId": "123456789012",
            "type": "AWS::Backup::BackupAccessPoint",
            "ARN": "arn:aws:backup:us-east-1:123456789012:accesspoint/my-access-point"
        },
        {
            "type": "AWS::S3::Object",
            "ARN": "arn:aws:s3:us-east-1:123456789012:accesspoint/my-access-point/
object/path/to/my-file.txt"
        },
        {
```

```
            "accountId": "123456789012",
            "type": "AWS::S3::AccessPoint",
            "ARN": "arn:aws:s3:us-east-1:123456789012:accesspoint/my-access-point"
        }
    ]
```

Control plane operations for backup access points (such as `CreateBackupAccessPoint` and `DeleteBackupAccessPoint`) are logged as AWS Backup events. For more information, see Monitoring AWS Backup in the *AWS Backup Developer Guide*.

For more information about S3 Server Access Logs, see Logging requests with server access logging. For more information about AWS CloudTrail, see What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

Creating an access point

You can create S3 access points by using the AWS Management Console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API. Access points are named network endpoints that are attached to a data source such as a bucket, Amazon FSx for ONTAP volume, or Amazon FSx for OpenZFS volume.

By default, you can create up to 10,000 access points per Region for each of your AWS accounts. If you need more than 10,000 access points for a single account in a single Region, you can request a service quota increase. For more information about service quotas and requesting an increase, see AWS Service Quotas in the *AWS General Reference*.

Topics

- Creating access points with S3 buckets
- Creating access points with Amazon FSx
- Creating access points with AWS Backup
- Creating access points restricted to a virtual private cloud
- Managing public access to access points for general purpose buckets

Creating access points with S3 buckets

An access point is associated with exactly one Amazon S3 general purpose bucket. If you want to use a bucket in your AWS account, you must first create a bucket. For more information about creating buckets, see Creating, configuring, and working with Amazon S3 general purpose buckets.

You can also create a cross-account access point that's associated with a bucket in another AWS account, as long as you know the bucket name and the bucket owner's account ID. However, creating cross-account access points doesn't grant you access to data in the bucket until you are granted permissions from the bucket owner. The bucket owner must grant the access point owner's account (your account) access to the bucket through the bucket policy. For more information, see Granting permissions for cross-account access points.

Using the S3 console

To create an access point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create an access point. The access point must be created in the same Region as the associated bucket.
3. In the left navigation pane, choose **Access Points**.
4. On the **Access Points** page, choose **Create access point**.
5. In the **Access point name** field, enter the name for the access point. For more information about naming access points, see Naming rules for access points.
6. For **Data source**, specify the S3 bucket that you want to use with the access point.

   To use a bucket in your account, choose **Choose a bucket in this account**, and enter or browse for the bucket name.

   To use a bucket in a different AWS account, choose **Specify a bucket in another account**, and enter the AWS account ID and name of the bucket. If you're using a bucket in a different AWS account, the bucket owner must update the bucket policy to authorize requests from the access point. For an example bucket policy, see Granting permissions for cross-account access points.

   > **Note**
   >
   > For information about using an FSx for OpenZFS volume as a data source, see Creating access points with Amazon FSx.

7. Choose a **Network origin**, either **Internet** or **virtual private cloud (VPC)**. If you choose **virtual private cloud (VPC)**, enter the **VPC ID** that you want to use with the access point.

For more information about network origins for access points, see Creating access points restricted to a virtual private cloud.

8. Under **Block Public Access settings for this Access Point**, select the block public access settings that you want to apply to the access point. All block public access settings are enabled by default for new access points. We recommend that you keep all settings enabled unless you know that you have a specific need to disable any of them.

   > **Note**
   >
   > After you create an access point, you can't change its block public access settings.

   For more information about using Amazon S3 Block Public Access with access points, see Managing public access to access points for general purpose buckets.

9. (Optional) Under **Access Point policy - *optional***, specify the access point policy. Before you save your policy, make sure to resolve any security warnings, errors, general warnings, and suggestions. For more information about specifying an access point policy, see Policy examples for access points.

10. Choose **Create access point**.

Using the AWS CLI

The following example command creates an access point named *example-ap* for the bucket *amzn-s3-demo-bucket* in the account *111122223333*. To create the access point, you send a request to Amazon S3 that specifies the following:

- The access point name. For information about naming rules, see the section called "Naming rules for access points".
- The name of the bucket that you want to associate the access point with.
- The account ID for the AWS account that owns the access point.

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket amzn-s3-demo-bucket
```

When you're creating an access point by using a bucket in a different AWS account, include the `--bucket-account-id` parameter. The following example command creates an access point in the AWS account *111122223333*, using the bucket *amzn-s3-demo-bucket2*, which is in the AWS account *444455556666*.

```
aws s3control create-access-point --name example-ap --account-id 111122223333 --
bucket amzn-s3-demo-bucket --bucket-account-id 444455556666
```

Using the REST API

You can use the REST API to create an access point. For more information, see CreateAccessPoint in the *Amazon Simple Storage Service API Reference*.

Creating access points with Amazon FSx

You can create and attach an access point to an FSx for OpenZFS volume using the Amazon FSx console, AWS CLI, or API. Once attached, you can use the S3 object APIs to access your file data. Your data continues to reside on the Amazon FSx file system and continues to be directly accessible for your existing workloads. You continue to manage your storage using all the FSx for OpenZFS storage management capabilities, including backups, snapshots, user and group quotas, and compression.

For instructions on creating an access point and attaching it to an FSx for OpenZFS volume see, Creating an access point in the *FSx for OpenZFS User Guide*.

Creating access points with AWS Backup

AWS Backup creates and attaches access points to S3 recovery points, providing read-only access to S3 backup data using S3 APIs. These access points are created through the AWS Backup console or the `CreateBackupAccessPoint` API, not through S3 access point APIs.

For instructions on creating a backup access point, see Backup access points in the *AWS Backup Developer Guide*.

Creating access points restricted to a virtual private cloud

When you create an access point you can choose to make the access point accessible from the internet, or you can specify that all requests made through that access point must originate from a specific virtual private cloud (VPC). An access point that's accessible from the internet is said to

have a network origin of `Internet`. It can be used from anywhere on the internet, subject to any other access restrictions in place for the access point, underlying data source, and related resources, such as the requested objects. An access point that's only accessible from a specified VPC has a network origin of VPC, and Amazon S3 rejects any request made to the access point that doesn't originate from that VPC.

> **Important**
>
> You can only specify an access point's network origin when you create the access point. After you create the access point, you can't change its network origin.

To restrict an access point to VPC-only access, you include the `VpcConfiguration` parameter with the request to create the access point. In the `VpcConfiguration` parameter, you specify the VPC ID that you want to be able to use the access point. If a request is made through the access point, the request must originate from the VPC or Amazon S3 will reject it.

You can retrieve an access point's network origin using the AWS CLI, AWS SDKs, or REST APIs. If an access point has a VPC configuration specified, its network origin is VPC. Otherwise, the access point's network origin is `Internet`.

Example: Create and restrict an access point to a VPC ID

The following example creates an access point named `example-vpc-ap` for bucket *amzn-s3-demo-bucket* in account 123456789012 that allows access only from the `vpc-1a2b3c` VPC. The example then verifies that the new access point has a network origin of VPC.

AWS CLI

```
aws s3control create-access-point --name example-vpc-ap --account-id 123456789012 --
bucket amzn-s3-demo-bucket --vpc-configuration VpcId=vpc-1a2b3c
```

```
aws s3control get-access-point --name example-vpc-ap --account-id 123456789012

{
    "Name": "example-vpc-ap",
    "Bucket": "amzn-s3-demo-bucket",
    "NetworkOrigin": "VPC",
    "VpcConfiguration": {
        "VpcId": "vpc-1a2b3c"
```

```
    },
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

To use an access point with a VPC, you must modify the access policy for your VPC endpoint. VPC endpoints allow traffic to flow from your VPC to Amazon S3. They have access control policies that control how resources within the VPC are allowed to interact with Amazon S3. Requests from your VPC to Amazon S3 only succeed through an access point if the VPC endpoint policy grants access to both the access point and the underlying bucket.

> **Note**
>
> To make resources accessible only within a VPC, make sure to create a private hosted zone for your VPC endpoint. To use a private hosted zone, modify your VPC settings so that the VPC network attributes `enableDnsHostnames` and `enableDnsSupport` are set to `true`.

For more information about VPC endpoint policies, see Using endpoint policies for Amazon S3 in the *VPC User Guide*.

For a tutorial on creating access points with VPC endpoints, see Managing Amazon S3 access with VPC endpoints and access points.

Example: Create and restrict an access point attached to an FSx for OpenZFS volume to a VPC ID

You can create an access point that is attached it to an FSx for OpenZFS volume using the Amazon FSx console, AWS CLI, or API. Once attached, you can use the S3 object APIs to access your file data from a specified VPC.

For instructions on creating and restricting an access point attached to an FSx for OpenZFS volume see, the Creating access points restricted to a virtual private cloud (VPC) in the *FSx for OpenZFS User Guide*.

Example: Create and restrict an access point attached to an FSX for ONTAP volume to a VPC ID

You can create an access point that is attached it to an FSx for ONTAP volume using the Amazon FSx console, AWS CLI, or API. Once attached, you can use the S3 object APIs to access your file data from a specified VPC.

For instructions on creating and restricting an access point attached to an FSx for ONTAP volume see, the *FSx for ONTAP User Guide*.

Managing public access to access points for general purpose buckets

Amazon S3 access points support independent *block public access* settings for each access point. When you create an access point, you can specify block public access settings that apply to that access point. For any request made through an access point, Amazon S3 evaluates the block public access settings for that access point, the underlying bucket, and the bucket owner's account. If any of these settings indicate that the request should be blocked, Amazon S3 rejects the request.

For more information about the S3 Block Public Access feature, see Blocking public access to your Amazon S3 storage.

Important

- All block public access settings are enabled by default for access points. You must explicitly disable any settings that you don't want during access point creation.
- You can not turn off any block public access settings when creating or using an access point attached to an Amazon FSx file system.
- After you create an access point, you can't change its block public access settings.

Example

Example: Create an access point with Custom Block Public Access Settings

This example creates an access point named `example-ap` for bucket *`amzn-s3-demo-bucket`* in account `123456789012` with non-default Block Public Access settings. The example then retrieves the new access point's configuration to verify its Block Public Access settings.

AWS CLI

```
aws s3control create-access-point --name example-ap --account-id
 123456789012 --bucket amzn-s3-demo-bucket--public-access-block-configuration
 BlockPublicAcls=false,IgnorePublicAcls=false,BlockPublicPolicy=true,RestrictPublicBuckets=t
```

```
aws s3control get-access-point --name example-ap --account-id 123456789012

{
    "Name": "example-ap",
    "Bucket": "amzn-s3-demo-bucket",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": false,
        "IgnorePublicAcls": false,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2019-11-27T00:00:00Z"
}
```

Managing your Amazon S3 access points for general purpose buckets

This section explains how to manage your Amazon S3 access points for general purpose buckets using the AWS Management Console, AWS Command Line Interface, or REST API. For information on managing access points attached to an FSx for OpenZFS volume, see Managing your Amazon S3 access points in the *FSx for OpenZFS User Guide*.

> **Note**
>
> You can only use access points to perform operations on objects. You can't use access points to perform other Amazon S3 operations, such as deleting buckets or creating S3 Replication configurations. For a complete list of S3 operations that support access points, see Access point compatibility.

Topics

- List your access points for general purpose buckets

- View details for your access point for general purpose buckets
- Delete your access point for a general purpose bucket

List your access points for general purpose buckets

This section explains how to list your access points for general purpose buckets using the AWS Management Console, AWS Command Line Interface, or REST API.

Using the S3 console

To list access points in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access Points**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage or use.

Using the AWS CLI

The following `list-access-points` example command shows how you can use the AWS CLI to list your access points.

The following command lists access points for AWS account *111122223333*.

```
aws s3control list-access-points --account-id 111122223333
```

The following command lists access points for AWS account *111122223333* that are attached to bucket *amzn-s3-demo-bucket*.

```
aws s3control list-access-points --account-id 111122223333 --bucket amzn-s3-demo-bucket
```

For more information and examples, see list-access-points in the *AWS CLI Command Reference*.

Using the REST API

You can use the REST API to list your access points. For more information, see ListAccessPoints in the *Amazon Simple Storage Service API Reference*.

View details for your access point for general purpose buckets

This section explains how to view details for your access point for a general purpose bucket using the AWS Management Console, AWS Command Line Interface, or REST API.

Using the S3 console

To view details for your access point in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access Points**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage or use.
6. Select the **Properties** tab to view the access point data source, account ID, AWS Region, creation date, network origin, S3 URI, ARN, and access point alias for the selected access point.
7. Select the **Permissions** tab to view the block public access settings and access point policy for the selected access point.

> **Note**
>
> You can't change any block public access settings for an access point after the access point is created.

Using the AWS CLI

The following `get-access-point` example command shows how you can use the AWS CLI to view details for your access point.

The following command lists details for the access point *my-access-point* for AWS account *111122223333* attached to S3 bucket *amzn-s3-demo-bucket*.

```
aws s3control get-access-point --name my-access-point --account-id 111122223333
```

Example output:

```
{
    "Name": "my-access-point",
    "Bucket": "amzn-s3-demo-bucket",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2016-08-29T22:57:52Z",
    "Alias": "my-access-point-u1ny6bhm7moymqx8cuon8o1g4mwikuse2a-s3alias",
    "AccessPointArn": "arn:aws:s3:AWS Region:111122223333:accesspoint/my-access-point",
    "Endpoints": {
        "ipv4": "s3-accesspoint.AWS Region.amazonaws.com",
        "fips": "s3-accesspoint-fips.AWS Region.amazonaws.com",
        "fips_dualstack": "s3-accesspoint-fips.dualstack.AWS Region.amazonaws.com",
        "dualstack": "s3-accesspoint.dualstack.AWS Region.amazonaws.com"
    },
    "BucketAccountId": "111122223333"
}
```

The following command lists details for the access point *example-fsx-ap* for AWS account *444455556666*. This access point is attached to an Amazon FSx file system.

```
aws s3control get-access-point --name example-fsx-ap --account-id 444455556666
```

Example output:

```
{
    "Name": "example-fsx-ap",
    "Bucket": "",
    "NetworkOrigin": "Internet",
    "PublicAccessBlockConfiguration": {
```

```
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    },
    "CreationDate": "2025-01-19T14:16:12Z",
    "Alias": "example-fsx-ap-qrqbyebjtsxorhhaa5exx6r3q7-ext-s3alias",
    "AccessPointArn": "arn:aws:s3:AWS Region:444455556666:accesspoint/example-fsx-ap",
    "Endpoints": {
        "ipv4": "s3-accesspoint.AWS Region.amazonaws.com",
        "fips": "s3-accesspoint-fips.AWS Region.amazonaws.com",
        "fips_dualstack": "s3-accesspoint-fips.dualstack.AWS Region.amazonaws.com",
        "dualstack": "s3-accesspoint.dualstack.AWS Region.amazonaws.com"
    },
    "DataSourceId": "arn:aws::fsx:AWS Region:444455556666:file-system/
fs-5432106789abcdef0/volume/vol-0123456789abcdef0",
    "DataSourceType": "FSX_OPENZFS"
}
```

For more information and examples, see get-access-point in the *AWS CLI Command Reference*.

Using the REST API

You can use the REST API to view details for your access point. For more information, see GetAccessPoint in the *Amazon Simple Storage Service API Reference*.

Delete your access point for a general purpose bucket

This section explains how to delete your access point for a general purpose bucket using the AWS Management Console, AWS Command Line Interface, or REST API.

Using the S3 console

To delete for your access points in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access Points**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.

5. Choose the name of the access point you want to manage or use.
6. From the **Access Point** page, select **Delete** to delete the access point you've selected.
7. To confirm deletion, type the name of the access point and choose **Delete**.

Using the AWS CLI

The following `delete-access-point` example command shows how you can use the AWS CLI to delete your access point.

The following command deletes the access point *`my-access-point`* for AWS account *`111122223333`*.

```
aws s3control delete-access-point --name my-access-point --account-id 111122223333
```

For more information and examples, see delete-access-point in the *AWS CLI Command Reference*.

Using the REST API

You can use the REST API to view details for your access point. For more information, see DeleteAccessPoint in the *Amazon Simple Storage Service API Reference*.

Using Amazon S3 access points for general purpose buckets

The following examples demonstrate how to use access points for general purpose buckets with compatible operations in Amazon S3.

> **Note**
>
> S3 automatically generate access point aliases for all access points and these aliases can be used anywhere a bucket name is used to perform object-level operations. For more information, see Access point aliases.

You can only use access points for general purpose buckets to perform operations on objects. You can't use access points to perform other Amazon S3 operations, such as modifying or deleting buckets. For a complete list of S3 operations that support access points, see Access point compatibility.

Topics

- List objects through an access point for a general purpose bucket
- Download an object through an access point for a general purpose bucket
- Configure access control lists (ACLs) through an access point for a general purpose bucket
- Upload an object through an access point for a general purpose bucket
- Add a tag-set through an access point for a general purpose bucket
- Delete an object through an access point for a general purpose bucket

List objects through an access point for a general purpose bucket

This section explains how to list your objects through an access point for a general purpose bucket using the AWS Management Console, AWS Command Line Interface, or REST API.

Using the S3 console

To list your objects through an access point in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access Points**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage or use.
6. Under the **Objects** tab, you can view the name of objects that you want to access through the access point. While you're using the access point, you can only perform the object operations that are allowed by the access point permissions.

> **Note**
>
> - The console view always shows all objects in the bucket. Using an access point as described in this procedure restricts the operations you can perform on those objects, but not whether you can see that they exist in the bucket.

- The AWS Management Console doesn't support using virtual private cloud (VPC) access points to access bucket resources. To access bucket resources from a VPC access point, use the AWS CLI, AWS SDKs, or Amazon S3 REST APIs.

Using the AWS CLI

The following `list-objects-v2` example command shows how you can use the AWS CLI to list your object through an access point.

The following command lists objects for AWS account *111122223333* using access point *my-access-point*.

```
aws s3api list-objects-v2 --bucket arn:aws:s3:AWS Region:111122223333:accesspoint/my-access-point
```

Note

S3 automatically generate access point aliases for all access points and these aliases can be used anywhere a bucket name is used to perform object-level operations. For more information, see Access point aliases.

For more information and examples, see list-access-points in the *AWS CLI Command Reference*.

Using the REST API

You can use the REST API to list your access points. For more information, see ListObjectsV2 in the *Amazon Simple Storage Service API Reference*.

Download an object through an access point for a general purpose bucket

This section explains how to download an object through an access point for a general purpose bucket using the AWS Management Console, AWS Command Line Interface, or REST API.

Using the S3 console

To download an object through an access point in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access Points**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage or use.
6. Under the **Objects** tab, select the name of object that you want to download.
7. Choose **Download**.

Using the AWS CLI

The following `get-object` example command shows how you can use the AWS CLI to download an object through an access point.

The following command downloads the object `puppy.jpg` for AWS account *111122223333* using access point *my-access-point*. You must include an `outfile`, which is a file name for the downloaded object, such as *my_downloaded_image.jpg*.

```
aws s3api get-object --bucket arn:aws:s3:AWS Region:111122223333:accesspoint/my-access-point --key puppy.jpg my_downloaded_image.jpg
```

> **Note**
>
> S3 automatically generate access point aliases for all access points and these aliases can be used anywhere a bucket name is used to perform object-level operations. For more information, see Access point aliases.

For more information and examples, see get-object in the *AWS CLI Command Reference*.

Using the REST API

You can use the REST API to download an object through an access point. For more information, see GetObject in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

You can use the AWS SDK for Python to download an object through an access point.

Python

In the following example, the file named *hello.txt* is downloaded for AWS account *111122223333* using the access point named *my-access-point*.

```
import boto3
s3 = boto3.client('s3')
s3.download_file('arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point',
 'hello.txt', '/tmp/hello.txt')
```

Configure access control lists (ACLs) through an access point for a general purpose bucket

This section explains how to configure ACLs through an access point for a general purpose bucket using the AWS Management Console, AWS Command Line Interface, or REST API. For more information about ACLs, see Access control list (ACL) overview.

Using the S3 console

To configure ACLs through an access point in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access Points**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage or use.
6. Under the **Objects** tab, select the name of the object you wish to configure an ACL for.
7. Under the **Permissions** tab, select **Edit** to configure the object ACL.

> **Note**
>
> Amazon S3 currently doesn't support changing an access point's block public access settings after the access point has been created.

Using the AWS CLI

The following `put-object-acl` example command shows how you can use the AWS CLI to configure access permissions through an access point using an ACL.

The following command applies an ACL to an existing object `puppy.jpg` through an access point owned by AWS account *111122223333*.

```
aws s3api put-object-acl --bucket arn:aws:s3:AWS Region:111122223333:accesspoint/my-access-point --key puppy.jpg --acl private
```

> **Note**
>
> S3 automatically generate access point aliases for all access points and these aliases can be used anywhere a bucket name is used to perform object-level operations. For more information, see Access point aliases.

For more information and examples, see put-object-acl in the *AWS CLI Command Reference*.

Using the REST API

You can use the REST API to configure access permissions through an access point using an ACL. For more information, see PutObjectAcl in the *Amazon Simple Storage Service API Reference*.

Upload an object through an access point for a general purpose bucket

This section explains how to upload an object through an access point for a general purpose bucket using the AWS Management Console, AWS Command Line Interface, or REST API.

Using the S3 console

To upload an object through an access point in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access Points**.

4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage or use.
6. Under the **Objects** tab, select **Upload**.
7. Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

   > **Note**
   >
   > The maximum size of a file that you can upload by using the Amazon S3 console is 160 GB. To upload a file larger than 160 GB, use the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

8. To change access control list permissions, choose **Permissions**.
9. Under **Access control list (ACL)**, edit the permissions.

   For information about object access permissions, see Using the S3 console to set ACL permissions for an object. You can grant read access to your objects to the public (everyone in the world) for all of the files that you're uploading. However, we recommend not changing the default setting for public read access. Granting public read access is applicable to a small subset of use cases, such as when buckets are used for websites. You can always change the object permissions after you upload the object.
10. To configure other additional properties, choose **Properties**.
11. Under **Storage class**, choose the storage class for the files that you're uploading.

    For more information about storage classes, see Understanding and managing Amazon S3 storage classes.
12. To update the encryption settings for your objects, under **Server-side encryption settings**, do the following.

    a. Choose **Specify an encryption key**.

    b. Under **Encryption settings**, choose **Use bucket settings for default encryption** or **Override bucket settings for default encryption**.

    c. If you chose **Override bucket settings for default encryption**, you must configure the following encryption settings.

- To encrypt the uploaded files by using keys that are managed by Amazon S3, choose **Amazon S3 managed key (SSE-S3)**.

  For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

- To encrypt the uploaded files by using keys stored in AWS Key Management Service (AWS KMS), choose **AWS Key Management Service key (SSE-KMS)**. Then choose one of the following options for **AWS KMS key**:

  - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose your **KMS key** from the list of available keys.

    Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.

  - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and then enter your KMS key ARN in the field that appears.

  - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

    For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

    Important

    You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that is not listed, you must enter your KMS key ARN. If you want to use a KMS key that is owned by a different account, you must first have permission to use the key and then you must enter the KMS key ARN.
    Amazon S3 supports only symmetric encryption KMS keys, and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

13. To use additional checksums, choose **On**. Then for **Checksum function**, choose the function that you would like to use. Amazon S3 calculates and stores the checksum value after it

receives the entire object. You can use the **Precalculated value** box to supply a precalculated value. If you do, Amazon S3 compares the value that you provided to the value that it calculates. If the two values do not match, Amazon S3 generates an error.

Additional checksums enable you to specify the checksum algorithm that you would like to use to verify your data. For more information about additional checksums, see Checking object integrity in Amazon S3.

14. To add tags to all of the objects that you are uploading, choose **Add tag**. Enter a tag name in the **Key** field. Enter a value for the tag.

    Object tagging gives you a way to categorize storage. Each tag is a key-value pair. Key and tag values are case sensitive. You can have up to 10 tags per object. A tag key can be up to 128 Unicode characters in length, and tag values can be up to 255 Unicode characters in length. For more information about object tags, see Tagging your objects.

15. To add metadata, choose **Add metadata**.

    a. Under **Type**, choose **System defined** or **User defined**.

       For system-defined metadata, you can select common HTTP headers, such as **Content-Type** and **Content-Disposition**. For a list of system-defined metadata and information about whether you can add the value, see System-defined object metadata. Any metadata starting with the prefix `x-amz-meta-` is treated as user-defined metadata. User-defined metadata is stored with the object and is returned when you download the object. Both the keys and their values must conform to US-ASCII standards. User-defined metadata can be as large as 2 KB. For more information about system-defined and user-defined metadata, see Working with object metadata.

    b. For **Key**, choose a key.

    c. Type a value for the key.

16. To upload your objects, choose **Upload**.

    Amazon S3 uploads your object. When the upload completes, you can see a success message on the **Upload: status** page.

Using the AWS CLI

The following `put-object` example command shows how you can use the AWS CLI to upload an object through an access point.

The following command uploads the object `puppy.jpg` for AWS account *111122223333* using access point *my-access-point*.

```
aws s3api put-object --bucket arn:aws:s3:AWS Region:111122223333:accesspoint/my-access-point --key puppy.jpg --body puppy.jpg
```

> **Note**
>
> S3 automatically generate access point aliases for all access points and access point aliases can be used anywhere a bucket name is used to perform object-level operations. For more information, see Access point aliases.

For more information and examples, see put-object in the *AWS CLI Command Reference*.

Using the REST API

You can use the REST API to upload an object through an access point. For more information, see PutObject in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

You can use the AWS SDK for Python to upload an object through an access point.

Python

In the following example, the file named *hello.txt* is uploaded for AWS account *111122223333* using the access point named *my-access-point*.

```
import boto3
s3 = boto3.client('s3')
s3.upload_file('/tmp/hello.txt', 'arn:aws:s3:us-east-1:111122223333:accesspoint/my-access-point', 'hello.txt')
```

Add a tag-set through an access point for a general purpose bucket

This section explains how to add a tag-set through an access point for a general purpose bucket using the AWS Management Console, AWS Command Line Interface, or REST API. For more information, see Tagging your objects.

Using the S3 console

To add a tag-set through an access point in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access Points**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage or use.
6. Under the **Objects** tab, select the name of the object you wish to add a tag-set to.
7. Under the **Properties** tab, find the **Tags** sub-header and choose **Edit**.
8. Review the objects listed, and choose **Add tag**.
9. Each object tag is a key-value pair. Enter a **Key** and a **Value**. To add another tag, choose **Add Tag**.

   You can enter up to 10 tags for an object.
10. Choose **Save changes**.

Using the AWS CLI

The following `put-object-tagging` example command shows how you can use the AWS CLI to add a tag-set through an access point.

The following command adds a tag-set for existing object `puppy.jpg` using access point *`my-access-point`*.

```
aws s3api put-object-tagging --bucket arn:aws:s3:AWS
 Region:111122223333:accesspoint/my-access-point --key puppy.jpg --tagging
 TagSet=[{Key="animal",Value="true"}]
```

> **Note**
>
> S3 automatically generate access point aliases for all access points and access point aliases can be used anywhere a bucket name is used to perform object-level operations. For more information, see Access point aliases.

For more information and examples, see put-object-tagging in the *AWS CLI Command Reference*.

Using the REST API

You can use the REST API to add a tag-set to an object through an access point. For more information, see PutObjectTagging in the *Amazon Simple Storage Service API Reference*.

Delete an object through an access point for a general purpose bucket

This section explains how to delete an object through an access point for a general purpose bucket using the AWS Management Console, AWS Command Line Interface, or REST API.

Using the S3 console

To delete an object or objects through an access point in your AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar on the top of the page, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to list access points for.
3. In the navigation pane on the left side of the console, choose **Access Points**.
4. (Optional) Search for access points by name. Only access points in your selected AWS Region will appear here.
5. Choose the name of the access point you want to manage or use.
6. Under the **Objects** tab, select the name of the object or objects you wish to delete.
7. Review the objects listed for deletion, and type *delete* in the confirmation box.
8. Choose **Delete objects**.

Using the AWS CLI

The following `delete-object` example command shows how you can use the AWS CLI to delete an object through an access point.

The following command deletes the existing object `puppy.jpg` using access point *`my-access-point`*.

```
aws s3api delete-object --bucket arn:aws:s3:AWS Region:111122223333:accesspoint/my-access-point --key puppy.jpg
```

> **Note**
>
> S3 automatically generate access point aliases for all access points and access point aliases can be used anywhere a bucket name is used to perform object-level operations. For more information, see Access point aliases.

For more information and examples, see delete-object in the *AWS CLI Command Reference*.

Using the REST API

You can use the REST API to delete an object through an access point. For more information, see DeleteObject in the *Amazon Simple Storage Service API Reference*.

Using tags with S3 Access Points for general purpose buckets

An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 Access Points. You can tag access points when you create them or manage tags on existing access points. For general information about tags, see Tagging for cost allocation or attribute-based access control (ABAC).

> **Note**
>
> There is no additional charge for using tags on access points beyond the standard S3 API request rates. For more information, see Amazon S3 pricing.

Common ways to use tags with access points

Attribute-based access control (ABAC) allows you to scale access permissions and grant access to access points based on their tags. For more information about ABAC in Amazon S3, see Using tags for ABAC.

ABAC for S3 Access Points

Amazon S3 Access Points support attribute-based access control (ABAC) using tags. Use tag-based condition keys in your AWS organizations, IAM, and Access Points policies. For enterprises, ABAC in Amazon S3 supports authorization across multiple AWS accounts.

In your IAM policies, you can control access to access points based on the access points's tags by using the following global condition keys:

- `aws:ResourceTag/key-name`

  > **Important**
  >
  > The `aws:ResourceTag` condition key can only be used for S3 actions performed via an access point ARN for general purpose buckets and covers the underlying access point tags only.

  - Use this key to compare the tag key-value pair that you specify in the policy with the key-value pair attached to the resource. For example, you could require that access to a resource is allowed only if the resource has the attached tag key `Dept` with the value `Marketing`. For more information, see Controlling access to AWS resources.
- `aws:RequestTag/key-name`
  - Use this key to compare the tag key-value pair that was passed in the request with the tag pair that you specify in the policy. For example, you could check whether the request includes the tag key `Dept` and that it has the value `Accounting`. For more information, see Controlling access during AWS requests. You can use this condition key to restrict which tag key-value pairs can be passed during the `TagResource` and `CreateAccessPoint` API operations.
- `aws:TagKeys`
  - Use this key to compare the tag keys in a request with the keys that you specify in the policy. We recommend that when you use policies to control access using tags, use the `aws:TagKeys` condition key to define what tag keys are allowed. For example policies and more information,

see Controlling access based on tag keys. You can create an access point with tags. To allow tagging during the `CreateAccessPoint` API operation, you must create a policy that includes both the `s3:TagResource` and `s3:CreateAccessPoint` actions. You can then use the `aws:TagKeys` condition key to enforce using specific tags in the `CreateAccessPoint` request.

- `s3:AccessPointTag/tag-key`
  - Use this condition key to grant permissions to specific data via access points using tags. When using `aws:ResourceTag/tag-key` in an IAM policy, both the access point as well as the bucket to which the access point points to are required to have the same tag as they are both considered during authorization. If you want to control access to your data specifically via the access-point tag only, you can use `s3:AccessPointTag/tag-key` condition key.

Example ABAC policies for access points

See the following example ABAC policies for Amazon S3 Access Points.

1.1 - IAM policy to create or modify buckets with specific tags

In this IAM policy, users or roles with this policy can only create access points if they tag the access points with the tag key `project` and tag value `Trinity` in the access points creation request. They can also add or modify tags on existing access points as long as the `TagResource` request includes the tag key-value pair `project:Trinity`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateAccessPointWithTags",
      "Effect": "Allow",
      "Action": [
        "s3:CreateAccessPoint",
        "s3:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/project": [
            "Trinity"
          ]
        }
```

```
      }
    }
  ]
}
```

1.2 - IAM policy to modify tags on existing resources maintaining tagging governence

In this IAM policy, IAM principals (users or roles) can modify tags on an access point only if the value of the access point's `project` tag matches the value of the principal's `project` tag. Only the four tags `project`, `environment`, `owner`, and `cost-center` specified in the `aws:TagKeys` condition keys are permitted for these access points. This helps enforce tag governance, prevents unauthorized tag modifications, and keeps the tagging schema consistent across your access points.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceTaggingRulesOnModification",
      "Effect": "Allow",
      "Action": [
        "s3:TagResource"
      ],
      "Resource": "arn:aws::s3:region:111122223333:accesspoint/my-access-point",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
        },
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "project",
            "environment",
            "owner",
            "cost-center"
          ]
        }
      }
    }
  ]
}
```

1.3 - Using the s3:AccessPointTag condition key

In this IAM policy, the condition statement allows access to the bucket's data if the access point has the tag key `Environment` and tag value `Production`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToSpecificAccessPoint",
      "Effect": "Allow",
      "Action": "*",
      "Resource": "arn:aws::s3:region:111122223333:accesspoint/my-access-point",
      "Condition": {
        "StringEquals": {
          "s3:AccessPointTag/Environment": "Production"
        }
      }
    }
  ]
}
```

1.4 - Using a bucket delegate policy

In Amazon S3, you can delegate access to or control of your S3 bucket policy to another AWS account or to a specific AWS Identity and Access Management (IAM) user or role in the other account. The delegate bucket policy grants this other account, user, or role permission to your bucket and its objects. For more information, see Permission delegation.

If you use a delegate bucket policy, you can use an IAM policy like the following. The condition statement allows access to the bucket\'s data if the access point has the tag key `Environment` and tag value `Production`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessToSpecificAccessPoint",
      "Effect": "Allow",
      "Action": "*",
      "Resource": "arn:aws::s3:region:111122223333:accesspoint/my-access-point",
      "Condition": {
```

```
        "StringEquals": {
          "s3:AccessPointTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Working with tags for access points for general purpose buckets

You can add or manage tags for access points using the Amazon S3 Console, the AWS Command Line Interface (CLI), the AWS SDKs, or using the S3 APIs: TagResource, UntagResource, and ListTagsForResource. For more information, see:

Topics

- Creating access points with tags
- Adding a tag to an access point
- Viewing access point tags
- Deleting a tag from an access point

Creating access points with tags

You can tag access points when you create them. There is no additional charge for using tags on access points beyond the standard S3 API request rates. For more information, see Amazon S3 pricing. For more information about tagging access points, see Using tags with S3 Access Points for general purpose buckets.

Permissions

To create an access point with tags, you must have the following permissions:

- `s3:CreateBucket`
- `s3:TagResource`

Troubleshooting errors

If you encounter an error when attempting to create an access point with tags, you can do the following:

- Verify that you have the required Permissions to create the access point and add a tag to it.
- Check your IAM user policy for any attribute-based access control (ABAC) conditions. You may be required to label your access points only with specific tag keys and values. For more information, see Using tags for attribute-based access control (ABAC).

Steps

You can create an access point with tags applied by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To create an access point with tags using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points (General Purpose Buckets)**.
3. Choose **create access point** to create a new access point.
4. On the **Create access point** page, **Tags** is an option when creating a new access point.
5. Enter a name for the access point. For more information, see Access points naming rules, restrictions, and limitations.
6. Choose **Add new Tag** to open the **Tags** editor and enter a tag key-value pair. The tag key is required, but the value is optional.
7. To add another tag, select **Add new Tag** again. You can enter up to 50 tag key-value pairs.
8. After you complete specifying the options for your new access point, choose **Create access point**.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to create an access point with tags by using the AWS SDK for Java 2.x. To use the command replace the *user input placeholders* with your own information.

```
CreateAccessPointRequest createAccessPointRequest =
 CreateAccessPointRequest.builder()
                .accountId(111122223333)
```

```
                .name(my-access-point)
                .bucket(amzn-s3-demo-bucket)

.tags(Collections.singletonList(Tag.builder().key("key1").value("value1").build()))
                .build();
awss3Control.createAccessPoint(createAccessPointRequest);
```

Using the REST API

For information about the Amazon S3 REST API support for creating an access point with tags, see the following section in the *Amazon Simple Storage Service API Reference*:

- CreateAccessPoint

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to create an access point with tags by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

Request:

```
aws s3control create-access-point --name my-access-point \
--bucket amzn-s3-demo-bucket \
--account-id 111122223333 \ --profile personal \
--tags [{Key=key1,Value=value1},{Key=key2,Value=value2}] \
--region region
```

Adding a tag to an access point

You can add tags to Amazon S3 Access Points and modify these tags. There is no additional charge for using tags on access points beyond the standard S3 API request rates. For more information, see Amazon S3 pricing. For more information about tagging access points, see Using tags with S3 Access Points for general purpose buckets.

Permissions

To add a tag to an access point, you must have the following permission:

- `s3:TagResource`

Troubleshooting errors

If you encounter an error when attempting to add a tag to an access point, you can do the following:

- Verify that you have the required Permissions to add a tag to an access point.
- If you attempted to add a tag key that starts with the AWS reserved prefix `aws:`, change the tag key and try again.

Steps

You can add tags to access points by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To add tags to an access point using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points (General Purpose Buckets)**.
3. Choose the access point name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and choose **Add new Tag**.
6. This opens the **Add Tags** page. You can enter up to 50 tag key value pairs.
7. If you add a new tag with the same key name as an existing tag, the value of the new tag overrides the value of the existing tag.
8. You can also edit the values of existing tags on this page.
9. After you have added the tag(s), choose **Save changes**.

Using the AWS SDKs

SDK for Java 2.x

> This example shows you how to add tags to an access point by using the AWS SDK for Java 2.x. To use the command replace the *`user input placeholders`* with your own information.

```
TagResourceRequest tagResourceRequest =
 TagResourceRequest.builder().resourceArn(arn:aws::s3:region:111122223333:accesspoint/
my-access-point/*)
.accountId(111122223333)
.tags(List.of(Tag.builder().key("key1").value("value1").build(),
Tag.builder().key("key2").value("value2").build()))
.build();
awss3Control.tagResource(tagResourceRequest);
```

Using the REST API

For information about the Amazon S3 REST API support for adding tags to an access point, see the following section in the *Amazon Simple Storage Service API Reference*:

- TagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to add tags to an access point by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

Request:

```
aws s3control tag-resource \
--account-id 111122223333 \
--resource-arn arn:aws::s3:region:111122223333:accesspoint/my-access-point/* \
--tags "Key=key1,Value=value1"
```

Response:

```
{
  "ResponseMetadata": {
      "RequestId": "EXAMPLE123456789",
      "HTTPStatusCode": 200,
      "HTTPHeaders": {
          "date": "Wed, 19 Jun 2025 10:30:00 GMT",
          "content-length": "0"
      },
```

```
      "RetryAttempts": 0
  }
}
```

Viewing access point tags

You can view or list tags applied to access points. For more information about tags, see Using tags with S3 Access Points for general purpose buckets.

Permissions

To view tags applied to an access point, you must have the following permission:

- `s3:ListTagsForResource`

Troubleshooting errors

If you encounter an error when attempting to list or view the tags of an access point, you can do the following:

- Verify that you have the required Permissions to view or list the tags of the access point.

Steps

You can view tags applied to access points by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To view tags applied to an access point using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points (General Purpose Buckets)**.
3. Choose the access point name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section to view all of the tags applied to the access point.
6. The **Tags** section shows the **User-defined tags** by default. You can select the **AWS-generated tags** tab to view tags applied to your access point by AWS services.

Using the AWS SDKs

This section provides an example of how to view tags applied to an access point by using the AWS SDKs.

SDK for Java 2.x

This example shows you how to view tags applied to an access point by using the AWS SDK for Java 2.x.

```
ListTagsForResourceRequest listTagsForResourceRequest = ListTagsForResourceRequest
.builder().resourceArn(arn:aws::s3:region:111122223333:accesspoint/my-access-point/
*)
                .accountId(111122223333).build();
awss3Control.listTagsForResource(listTagsForResourceRequest);
```

Using the REST API

For information about the Amazon S3 REST API support for viewing the tags applied to an access point, see the following section in the *Amazon Simple Storage Service API Reference*:

- ListTagsforResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to view tags applied to an access point. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws s3control list-tags-for-resource \
--account-id 111122223333 \
--resource-arn arn:aws::s3:region:444455556666:bucket/prefix--use1-az4--x-s3 \
```

Response - tags present:

```
{
```

```
  "Tags": [
      {
          "Key": "MyKey1",
          "Value": "MyValue1"
      },
      {
          "Key": "MyKey2",
          "Value": "MyValue2"
      },
      {
          "Key": "MyKey3",
          "Value": "MyValue3"
      }
  ]
}
```

Response - no tags present:

```
{
  "Tags": []
}
```

Deleting a tag from an access point

You can remove tags from Amazon S3 Access Points. An AWS tag is a key-value pair that holds metadata about resources, in this case Access Points. For more information about tags, see Using tags with S3 Access Points for general purpose buckets.

> **Note**
>
> If you delete a tag and later learn that it was being used to track costs or for access control, you can add the tag back to the access point.

Permissions

To delete a tag from an access point, you must have the following permission:

- `s3:UntagResource`

Troubleshooting errors

If you encounter an error when attempting to delete a tag from an access point, you can do the following:

- Verify that you have the required Permissions to delete a tag from an access point.

Steps

You can delete tags from access points by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To delete tags from an access point using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Points (General Purpose Buckets)**.
3. Choose the access point name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and select the checkbox next to the tag or tags that you would like to delete.
6. Choose **Delete**.
7. The **Delete user-defined tags** pop-up appears and asks you to confirm the deletion of the tag or tags you selected.
8. Choose **Delete** to confirm.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to delete tags from an access point by using the AWS SDK for Java 2.x. To use the command replace the *user input placeholders* with your own information.

```
UntagResourceRequest tagResourceRequest = UntagResourceRequest.builder()
                .resourceArn(arn:aws::s3:region:111122223333:accesspoint/my-access-
point/*)
```

```
                .accountId(111122223333)
                .tagKeys(List.of("key1", "key2")).build();
awss3Control.untagResource(tagResourceRequest);
```

Using the REST API

For information about the Amazon S3 REST API support for deleting tags from an access point, see the following section in the *Amazon Simple Storage Service API Reference*:

- UnTagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to delete tags from an access point by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws s3control untag-resource \
--account-id 111122223333 \
--resource-arn arn:aws::s3:region:444455556666:access-point/my-access-point \
--tag-keys "tagkey1" "tagkey2"


  aws s3control untag-resource \
--account-id 111122223333 \
--resource-arn arn:aws::s3:region:444455556666:accesspointmy-access-point/* \
--tag-keys "key1" "key2"
```

Response:

```
{
  "ResponseMetadata": {
    "RequestId": "EXAMPLE123456789",
    "HTTPStatusCode": 204,
    "HTTPHeaders": {
        "date": "Wed, 19 Jun 2025 10:30:00 GMT",
        "content-length": "0"
    },
```

```
    "RetryAttempts": 0
  }
}
```

Managing access with S3 Access Grants

To adhere to the principle of least privilege, you define granular access to your Amazon S3 data based on applications, personas, groups, or organizational units. You can use various approaches to achieve granular access to your data in Amazon S3, depending on the scale and complexity of the access patterns.

The simplest approach for managing access to small-to-medium numbers of datasets in Amazon S3 by AWS Identity and Access Management (IAM) principals is to define IAM permission policies and S3 bucket policies. This strategy works, so long as the necessary policies fit within the policy size limits of S3 bucket policies (20 KB) and IAM policies (5 KB), and within the number of IAM principals allowed per account.

As your number of datasets and use cases scales, you might require more policy space. An approach that offers significantly more space for policy statements is to use S3 Access Points as additional endpoints for S3 buckets, because each access point can have its own policy. You can define quite granular access control patterns, because you can have thousands of access points per AWS Region per account, with a policy up to 20 KB in size for each access point. Although S3 Access Points increases the amount of policy space available, it requires a mechanism for clients to discover the right access point for the right dataset.

A third approach is to implement an IAM session broker pattern, in which you implement access-decision logic and dynamically generate short-term IAM session credentials for each access session. While the IAM session broker approach supports arbitrarily dynamic permissions patterns and scales effectively, you must build the access-pattern logic.

Instead of using these approaches, you can use S3 Access Grants to manage access to your Amazon S3 data. S3 Access Grants provides a simplified model for defining access permissions to data in Amazon S3 by prefix, bucket, or object. In addition, you can use S3 Access Grants to grant access to both IAM principals and directly to users or groups from your corporate directory.

You commonly define permissions to data in Amazon S3 by mapping users and groups to datasets. You can use S3 Access Grants to define direct access mappings of S3 prefixes to users and roles within Amazon S3 buckets and objects. With the simplified access scheme in S3 Access Grants, you

can grant read-only, write-only, or read-write access on a per-S3-prefix basis to both IAM principals and directly to users or groups from a corporate directory. With these S3 Access Grants capabilities, applications can request data from Amazon S3 on behalf of the application's current authenticated user.

When you integrate S3 Access Grants with the trusted identity propagation feature of AWS IAM Identity Center, your applications can make requests to AWS services (including S3 Access Grants) directly on behalf of an authenticated corporate directory user. Your applications no longer need to first map the user to an IAM principal. Furthermore, because end-user identities are propagated all the way to Amazon S3, auditing which user accessed which S3 object is simplified. You no longer need to reconstruct the relationship between different users and IAM sessions. When you're using S3 Access Grants with IAM Identity Center trusted identity propagation, each AWS CloudTrail data event for Amazon S3 contains a direct reference to the end user on whose behalf the data was accessed.

Trusted identity propagation is an AWS IAM Identity Center feature that administrators of connected AWS services can use to grant and audit access to service data. Access to this data is based on user attributes such as group associations. Setting up trusted identity propagation requires collaboration between the administrators of connected AWS services and the IAM Identity Center administrators. For more information, see Prerequisites and considerations.

For more information about S3 Access Grants, see the following topics.

Topics

- S3 Access Grants concepts
- S3 Access Grants and corporate directory identities
- Getting started with S3 Access Grants
- Working with S3 Access Grants instances
- Working with S3 Access Grants locations
- Working with grants in S3 Access Grants
- Getting S3 data using access grants
- S3 Access Grants cross-account access
- Managing tags for S3 Access Grants
- S3 Access Grants limitations
- S3 Access Grants integrations

S3 Access Grants concepts

S3 Access Grants workflow

The S3 Access Grants workflow is:

1. Create an S3 Access Grants instance. See Working with S3 Access Grants instances.
2. Within your S3 Access Grants instance, register locations in your Amazon S3 data, and map these locations to AWS Identity and Access Management (IAM) roles. See Register a location.
3. Create grants for grantees, which give grantees access to your S3 resources. See Working with grants in S3 Access Grants.
4. The grantee requests temporary credentials from S3 Access Grants. See Request access to Amazon S3 data through S3 Access Grants.
5. The grantee accesses the S3 data using those temporary credentials. See Accessing S3 data using credentials vended by S3 Access Grants.

For more information, see Getting started with S3 Access Grants.

S3 Access Grants instances

An *S3 Access Grants instance* is a logical container for individual *grants*. When you create an S3 Access Grants instance, you must specify an AWS Region. Each AWS Region in your AWS account can have one S3 Access Grants instance. For more information, see Working with S3 Access Grants instances.

If you want to use S3 Access Grants to grant access to user and group identities from your corporate directory, you must also associate your S3 Access Grants instance with an AWS IAM Identity Center instance. For more information, see S3 Access Grants and corporate directory identities.

A newly created S3 Access Grants instance is empty. You must register a location in the instance, which can be the S3 default path (`s3://`), a bucket, or a prefix within a bucket. After you register at least one location, you can create access grants that give access to data in this registered location.

Locations

An S3 Access Grants *location* maps buckets or prefixes to an AWS Identity and Access Management (IAM) role. S3 Access Grants assumes this IAM role to vend temporary credentials

to the grantee that's accessing that particular location. You must first register at least one location in your S3 Access Grants instance before you can create an access grant.

We recommend that you register the default location (`s3://`) and map it to an IAM role. The location at the default S3 path (`s3://`) covers access to all of your S3 buckets in the AWS Region of your account. When you create an access grant, you can narrow the grant scope to a bucket, a prefix, or an object within the default location.

More complex access-management use cases might require you to register more than the default location. Some examples of such use cases are:

- Suppose that the *amzn-s3-demo-bucket* is a registered location in your S3 Access Grants instance with an IAM role mapped to it, but this IAM role is denied access to a particular prefix within the bucket. In this case, you can register the prefix that the IAM role does not have access to as a separate location and map that location to a different IAM role with the necessary access.
- Suppose that you want to create grants that restrict access to only the users within a virtual private cloud (VPC) endpoint. In this case, you can register a location for a bucket in which the IAM role restricts access to the VPC endpoint. Later, when a grantee asks S3 Access Grants for credentials, S3 Access Grants assumes the location's IAM role to vend the temporary credentials. This credential will deny access to the specific bucket unless the caller is within the VPC endpoint. This deny permission is applied in addition to the regular READ, WRITE, or READWRITE permission specified in the grant.

If your use case requires you to register multiple locations in your S3 Access Grants instance, you can register any of the following:

- The default S3 location (`s3://`)
- A bucket (for example, *amzn-s3-demo-bucket*) or multiple buckets
- A bucket and a prefix (for example, *amzn-s3-demo-bucket*`/prefix*`) or multiple prefixes

For the maximum number of locations that you can register in your S3 Access Grants instance, see S3 Access Grants limitations. For more information about registering an S3 Access Grants location, see Register a location.

After you register the first location in your S3 Access Grants instance, your instance still does not have any individual access grants in it. So, no access has been granted yet to any of your S3 data. You can now create access grants to give access. For more information about creating grants, see Working with grants in S3 Access Grants.

Grants

An individual *grant* in an S3 Access Grants instance allows a specific identity—an IAM principal, or a user or group in a corporate directory—to get access within a location that is registered in your S3 Access Grants instance.

When you create a grant, you don't have to grant access to the entire registered location. You can narrow the grant's scope of access within a location. If the registered location is the default S3 path (`s3://`), you are required to narrow the scope of the grant to a bucket, a prefix within a bucket, or a specific object. If the registered location of the grant is a bucket or a prefix, then you can give access to the entire bucket or prefix, or you can optionally narrow the scope of the grant to a prefix, subprefix, or an object.

In the grant, you also set the access level of the grant to READ, WRITE, or READWRITE. Suppose you have a grant that gives the corporate directory group `01234567-89ab-cdef-0123-456789abcdef` READ access to the bucket `s3://`*`amzn-s3-demo-bucket`*`/projects/items/*`. Users in this group can have READ access to every object that has an object key name which starts with the prefix `projects/items/` in the bucket named *`amzn-s3-demo-bucket`*.

For the maximum number of grants that you can create in your S3 Access Grants instance, see S3 Access Grants limitations. For more information about creating grants, see Create grants.

S3 Access Grants temporary credentials

After you create a grant, an authorized application that utilizes the identity specified in the grant can request *just-in-time access credentials*. To do this, the application calls the GetDataAccess S3 API operation. Grantees can use this API operation to request access to the S3 data you have shared with them.

The S3 Access Grants instance evaluates the `GetDataAccess` request against the grants that it has. If there is a matching grant for the requestor, S3 Access Grants assumes the IAM role that's associated with the registered location of the matching grant. S3 Access Grants scopes the permissions of the temporary credentials to access only the S3 bucket, prefix, or object that's specified by the grant's scope.

The expiration time of the temporary access credentials defaults to 1 hour, but you can set it to any value from 15 minutes to 12 hours. See the maximum duration session in the AssumeRole API reference.

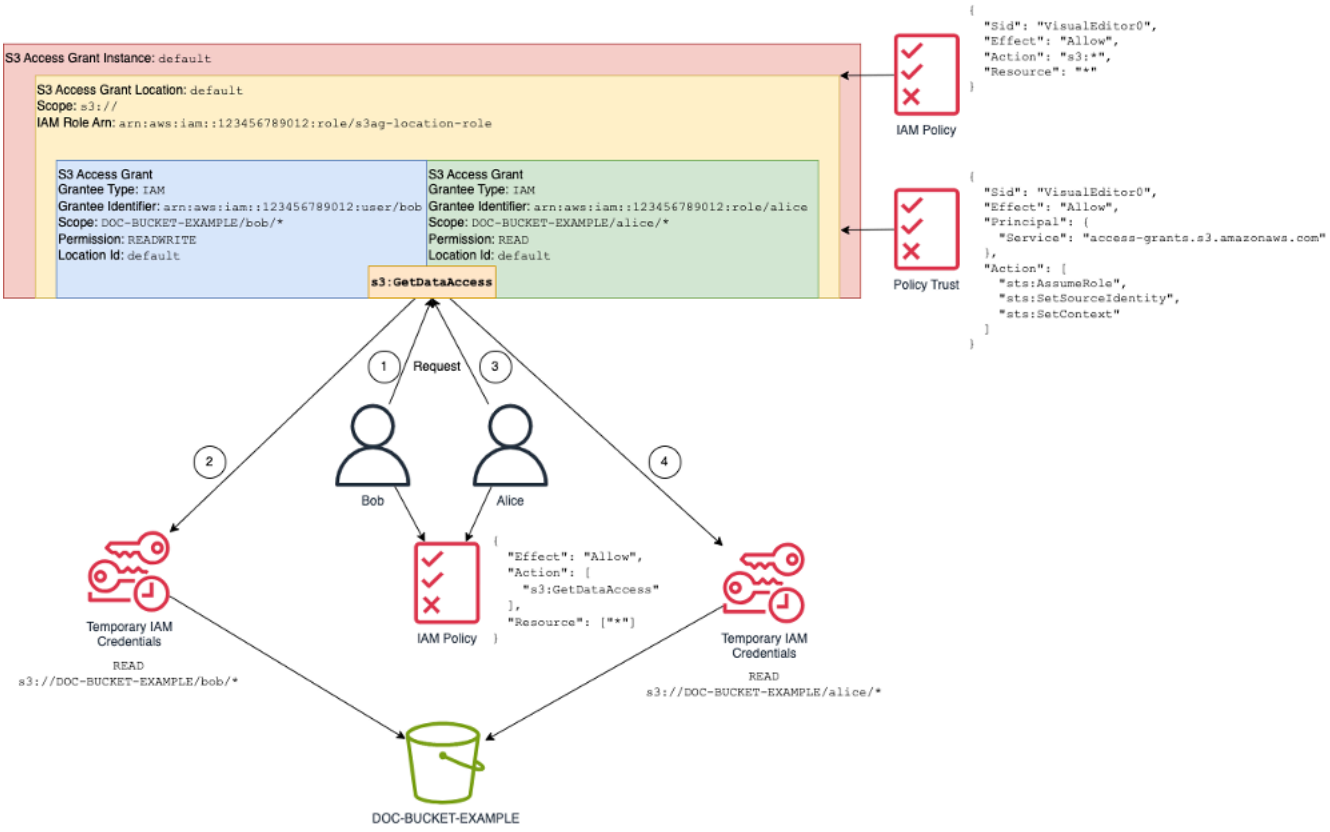
How it works

In the following diagram, a default Amazon S3 location with the scope `s3://` is registered with the IAM role `s3ag-location-role`. This IAM role has permissions to perform Amazon S3 actions within the account when its credentials are obtained through S3 Access Grants.

Within this location, two individual access grants are created for two IAM users. The IAM user Bob is granted both `READ` and `WRITE` access on the `bob/` prefix in the `DOC-BUCKET-EXAMPLE` bucket. Another IAM role, Alice, is granted only `READ` access on the `alice/` prefix in the `DOC-BUCKET-EXAMPLE` bucket. A grant, colored in blue, is defined for Bob to access the prefix `bob/` in the `DOC-BUCKET-EXAMPLE` bucket. A grant, colored in green, is defined for Alice to access the prefix `alice/` in the `DOC-BUCKET-EXAMPLE` bucket.

When it's time for Bob to `READ` data, the IAM role that's associated with the location that his grant is in calls the S3 Access Grants GetDataAccess API operation. If Bob tries to `READ` any S3 prefix or object that starts with `s3://DOC-BUCKET-EXAMPLE/bob/*`, the `GetDataAccess` request returns a set of temporary IAM session credentials with permission to `s3://DOC-BUCKET-EXAMPLE/bob/*`. Similarly, Bob can `WRITE` to any S3 prefix or object that starts with `s3://DOC-BUCKET-EXAMPLE/bob/*`, because the grant also allows that.

Similarly, Alice can `READ` anything that starts with `s3://DOC-BUCKET-EXAMPLE/alice/`. However, if she tries to `WRITE` anything to any bucket, prefix, or object in `s3://`, she will get an Access Denied (403 Forbidden) error, because there is no grant that gives her `WRITE` access to any data. In addition, if Alice requests any level of access (`READ` or `WRITE`) to data outside of `s3://DOC-BUCKET-EXAMPLE/alice/`, she will again receive an Access Denied error.



This pattern scales to a high number of users and buckets and simplifies management of those permissions. Rather than editing potentially large S3 bucket policies every time you want to add or remove an individual user-prefix access relationship, you can add and remove individual, discrete grants.

S3 Access Grants and corporate directory identities

You can use Amazon S3 Access Grants to grant access to AWS Identity and Access Management (IAM) principals (users or roles), both in the same AWS account and in others. However, in many cases, the entity accessing the data is an end user from your corporate directory. Instead of granting access to IAM principals, you can use S3 Access Grants to grant access directly to your corporate users and groups. With S3 Access Grants, you no longer need to map your corporate identities to intermediate IAM principals in order to access your S3 data through your corporate applications.

This new functionality—support for using end-user identities access to data—is provided by associating your S3 Access Grants instance with an AWS IAM Identity Center instance. IAM Identity Center supports standards-based identity providers and is the hub in AWS for any services or features, including S3 Access Grants, that support end-user identities. IAM Identity Center provides

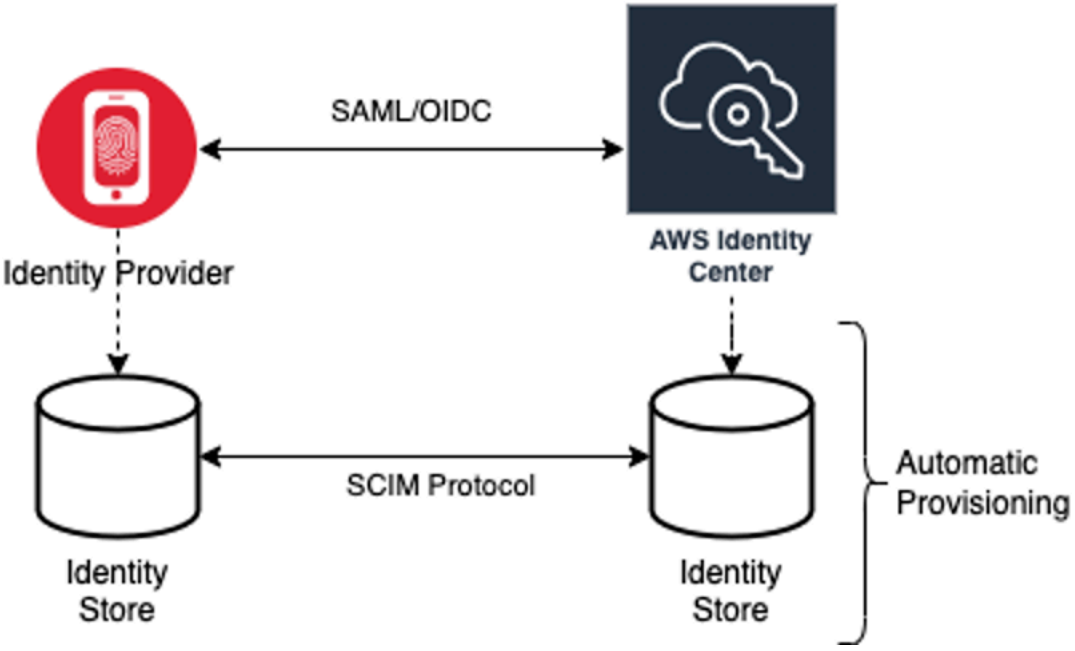
authentication support for corporate identities through its trusted identity propagation feature. For more information, see Trusted identity propagation across applications.

To get started with workforce identity support in S3 Access Grants, as a prerequisite, you start in IAM Identity Center by configuring identity provisioning between your corporate identity provider and IAM Identity Center. IAM Identity Center supports corporate identity providers such as Okta, Microsoft Entra ID (formerly Azure Active Directory), or any other external identity provider (IdP) that supports the System for Cross-domain Identity Management (SCIM) protocol. When you connect IAM Identity Center to your IdP and enable automatic provisioning, the users and groups from your IdP are synchronized into the identity store in IAM Identity Center. After this step, IAM Identity Center has its own view of your users and groups, so that you can refer to them by using other AWS services and features, such as S3 Access Grants. For more information about configuring IAM Identity Center automatic provisioning, see Automatic provisioning in the *AWS IAM Identity Center User Guide*.

IAM Identity Center is integrated with AWS Organizations so that you can centrally manage permissions across multiple AWS accounts without configuring each of your accounts manually. In a typical organization, your identity administrator configures one IAM Identity Center instance for the entire organization, as a single point of identity synchronization. This IAM Identity Center instance typically runs in a dedicated AWS account in your organization. In this common configuration, you can refer to user and group identities in S3 Access Grants from any AWS account in the organization.

However, if your AWS Organizations administrator hasn't yet configured a central IAM Identity Center instance, you can create a local one in the same account and AWS Region as your S3 Access Grants instance. If you have an IAM Identity Center instance configured in a different AWS Region, you can also replicate this instance to the same AWS Region as your S3 Access Grants instance. Such a configuration is more common for proof-of-concept or local development use cases. In all cases, the IAM Identity Center instance must be in the same AWS Region as the S3 Access Grants instance to which it will be associated.

In the following diagram of an IAM Identity Center configuration with an external IdP, the IdP is configured with SCIM to synchronize the identity store from the IdP to the identity store in IAM Identity Center.



To use your corporate directory identities with S3 Access Grants, do the following:

- Set up Automatic provisioning in IAM Identity Center to synchronize user and group information from your IdP into IAM Identity Center.
- Configure your external identity source within IAM Identity Center as a trusted token issuer. For more information, see Trusted identity propagation across applications in the *AWS IAM Identity Center User Guide*.
- Associate your S3 Access Grants instance with your IAM Identity Center instance. You can do this when you create your S3 Access Grants instance. If you've already created your S3 Access Grants instance, see Associate or disassociate your IAM Identity Center instance.

How directory identities can access S3 data

Suppose that you have corporate directory users who need to access your S3 data through a corporate application, for example, a document-viewer application, that is integrated with your external IdP (for example, Okta) to authenticate users. Authentication of the user in these applications is typically done through redirects in the user's web browser. Because users in the directory are not IAM principals, your application needs IAM credentials with which it can call the S3 Access Grants `GetDataAccess` API operation to get access credentials to S3 data on the users' behalf. Unlike IAM users and roles who get credentials themselves, your application needs a way to represent a directory user, who isn't mapped to an IAM role, so that the user can get data access through S3 Access Grants.

This transition, from authenticated directory user to an IAM caller that can make requests to S3 Access Grants on behalf of the directory user, is done by the application through the trusted token issuer feature of IAM Identity Center. The application, after authenticating the directory user, has an identity token from the IdP (for example, Okta) that represents the directory user according to Okta. The trusted token issuer configuration in IAM Identity Center enables the application to exchange this Okta token (the Okta tenant is configured as the "trusted issuer") for a different identity token from IAM Identity Center that will securely represent the directory user within AWS services. The data application will then assume an IAM role, providing the directory user's token from IAM Identity Center as additional context. The application can use the resulting IAM session to call S3 Access Grants. The token represents both the identity of the application (the IAM principal itself) as well as the directory user's identity.

The main step of this transition is the token exchange. The application performs this token exchange by calling the `CreateTokenWithIAM` API operation in IAM Identity Center. Of course, that too is an AWS API call and requires an IAM principal to sign it. The IAM principal that makes this request is typically an IAM role that's associated with the application. For example, if the application runs on Amazon EC2, the `CreateTokenWithIAM` request is typically performed by the IAM role that's associated with the EC2 instance on which the application runs. The result of a successful `CreateTokenWithIAM` call is a new identity token, which will be recognized within AWS services.

The next step, before the application can call `GetDataAccess` on the directory user's behalf, is for the application to obtain an IAM session that includes the directory user's identity. The application does this with an AWS Security Token Service (AWS STS) `AssumeRole` request that also includes the IAM Identity Center token for the directory user as additional identity context. This additional context is what enables IAM Identity Center to propagate the directory user's identity to the next step. The IAM role that the application assumes is the role that will need IAM permissions to call the `GetDataAccess` operation.

Having assumed the identity bearer IAM role with the IAM Identity Center token for the directory user as additional context, the application now has everything it needs to make a signed request to `GetDataAccess` on behalf of the authenticated directory user.

Token propagation is based on the following steps:

Create an IAM Identity Center application

First, create a new application in IAM Identity Center. This application will use a template that allows IAM Identity Center to identify which type of application settings that you can use. The

command to create the application requires you to provide the IAM Identity Center instance Amazon Resource Name (ARN), an application name, and the application provider ARN. The application provider is the SAML or OAuth application provider that the application will use to make calls to IAM Identity Center.

To use the following example command, replace the *user input placeholders* with your own information:

```
aws sso-admin create-application \
 --instance-arn "arn:aws:sso:::instance/ssoins-ssoins-1234567890abcdef" \
 --application-provider-arn "arn:aws:sso::aws:applicationProvider/custom" \
 --name MyDataApplication
```

Response:

```
{
   "ApplicationArn": "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d"
}
```

Create a trusted token issuer

Now that you have your IAM Identity Center application, the next step is to configure a trusted token issuer that will be used to exchange your `IdToken` values from your IdP with IAM Identity Center tokens. In this step you need to provide the following items:

- The identity provider issuer URL
- The trusted token issuer name
- The claim attribute path
- The identity store attribute path
- The JSON Web Key Set (JWKS) retrieval option

The claim attribute path is the identity provider attribute that will be used to map to the identity store attribute. Normally, the claim attribute path is the email address of the user, but you can use other attributes to perform the mapping.

Create a file called `oidc-configuration.json` with the following information. To use this file, replace the *user input placeholders* with your own information.

```
{
  "OidcJwtConfiguration":
     {
      "IssuerUrl": "https://login.microsoftonline.com/a1b2c3d4-abcd-1234-b7d5-
b154440ac123/v2.0",
      "ClaimAttributePath": "preferred_username",
      "IdentityStoreAttributePath": "userName",
      "JwksRetrievalOption": "OPEN_ID_DISCOVERY"
     }
}
```

To create the trusted token issuer, run the following command. To use this example command, replace the *user input placeholders* with your own information.

```
aws sso-admin create-trusted-token-issuer \
  --instance-arn "arn:aws:sso:::instance/ssoins-1234567890abcdef" \
  --name MyEntraIDTrustedIssuer \
  --trusted-token-issuer-type OIDC_JWT \
  --trusted-token-issuer-configuration file://./oidc-configuration.json
```

Response

```
{
  "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234"
}
```

Connect the IAM Identity Center application with the trusted token issuer

The trusted token issuer requires a few more configuration settings to work. Set the audience that the trusted token issuer will trust. The audience is the value inside the `IdToken` that's identified by the key and can be found in the identity provider settings. For example:

```
1234973b-abcd-1234-abcd-345c5a9c1234
```

Create a file named `grant.json` that contains the following content. To use this file, change the audience to match your identity provider settings and provide the trusted token issuer ARN that was returned by the previous command.

```
{
```

```
   "JwtBearer":
     {
       "AuthorizedTokenIssuers":
         [
           {
             "TrustedTokenIssuerArn": "arn:aws:sso::123456789012:trustedTokenIssuer/
ssoins-1234567890abcdef/tti-43b4a822-1234-1234-1234-a1b2c3d41234",
               "AuthorizedAudiences":
                 [
                   "1234973b-abcd-1234-abcd-345c5a9c1234"
                 ]
            }
         ]
     }
 }
```

Run the following example command. To use this command, replace the *`user input placeholders`* with your own information.

```
aws sso-admin put-application-grant \
  --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
  --grant-type "urn:ietf:params:oauth:grant-type:jwt-bearer" \
  --grant file://./grant.json \
```

This command sets the trusted token issuer with configuration settings to trust the audience in the `grant.json` file and link this audience with the application created in the first step for exchanging tokens of the type `jwt-bearer`. The string `urn:ietf:params:oauth:grant-type:jwt-bearer` is not an arbitrary string. It is a registered namespace in OAuth JSON Web Token (JWT) assertion profiles. You can find more information about this namespace in RFC 7523.

Next, use the following command to set up which scopes the trusted token issuer will include when exchanging `IdToken` values from your identity provider. For S3 Access Grants, the value for the `--scope` parameter is `s3:access_grants:read_write`.

```
aws sso-admin put-application-access-scope \
  --application-arn "arn:aws:sso::111122223333:application/ssoins-
ssoins-111122223333abcdef/apl-abcd1234a1b2c3d" \
  --scope "s3:access_grants:read_write"
```

The last step is to attach a resource policy to the IAM Identity Center application. This policy will allow your application IAM role to make requests to the API operation `sso-oauth:CreateTokenWithIAM` and receive the `IdToken` values from IAM Identity Center.

Create a file named `authentication-method.json` that contains the following content. Replace *123456789012* with your account ID.

```
{
   "Iam":
       {
         "ActorPolicy":
             {
                "Version": "2012-10-17",
                    "Statement":
                    [
                        {
                           "Effect": "Allow",
                            "Principal":
                            {
                              "AWS": "arn:aws:iam::123456789012:role/webapp"
                            },
                           "Action": "sso-oauth:CreateTokenWithIAM",
                            "Resource": "*"
                        }
                    ]
                }
            }
        }
```

To attach the policy to the IAM Identity Center application, run the following command:

```
aws sso-admin put-application-authentication-method \
   --application-arn "arn:aws:sso::123456789012:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d" \
   --authentication-method-type IAM \
   --authentication-method file://./authentication-method.json
```

This completes the configuration settings for using S3 Access Grants with directory users through a web application. You can test this setup directly in the application or you can call the `CreateTokenWithIAM` API operation by using the following command from an allowed IAM role in the IAM Identity Center application policy:

```
aws sso-oidc create-token-with-iam \
   --client-id "arn:aws:sso::123456789012:application/ssoins-ssoins-1234567890abcdef/
apl-abcd1234a1b2c3d"  \
   --grant-type urn:ietf:params:oauth:grant-type:jwt-bearer \
   --assertion IdToken
```

The response will be similar to this:

```
{
    "accessToken": "<suppressed long string to reduce space>",
    "tokenType": "Bearer",
    "expiresIn": 3600,
    "refreshToken": "<suppressed long string to reduce space>",
    "idToken": "<suppressed long string to reduce space>",
    "issuedTokenType": "urn:ietf:params:oauth:token-type:refresh_token",
    "scope": [
      "sts:identity_context",
      "s3:access_grants:read_write",
      "openid",
      "aws"
    ]
}
```

If you decode the `IdToken` value that is encoded with base64, you can see the key-value pairs in JSON format. The key `sts:identity_context` contains the value that your application needs to send in the `sts:AssumeRole` request to include the identity information of the directory user. Here is an example of the `IdToken` decoded:

```
{
    "aws:identity_store_id": "d-996773e796",
    "sts:identity_context": "AQoJb3JpZ2luX2VjEOTtl;<SUPRESSED>",
    "sub": "83d43802-00b1-7054-db02-f1d683aacba5",
    "aws:instance_account": "123456789012",
    "iss": "https://identitycenter.amazonaws.com/ssoins-1234567890abcdef",
    "sts:audit_context": "AQoJb3JpZ2luX2VjEOT<SUPRESSED>==",
    "aws:identity_store_arn": "arn:aws:identitystore::232642235904:identitystore/
d-996773e796",
    "aud": "abcd12344U0gi7n4Yyp0-WV1LWNlbnRyYWwtMQ",
    "aws:instance_arn": "arn:aws:sso:::instance/ssoins-6987d7fb04cf7a51",
    "aws:credential_id": "EXAMPLEHI5glPh40y9TpApJn8...",
    "act": {
```

```
       "sub": "arn:aws:sso::232642235904:trustedTokenIssuer/
ssoins-6987d7fb04cf7a51/43b4a822-1020-7053-3631-cb2d3e28d10e"
    },
    "auth_time": "2023-11-01T20:24:28Z",
    "exp": 1698873868,
    "iat": 1698870268
}
```

You can get the value from `sts:identity_context` and pass this information in an `sts:AssumeRole` call. The following is a CLI example of the syntax. The role to be assumed is a temporary role with permissions to invoke `s3:GetDataAccess`.

```
aws sts assume-role \
   --role-arn "arn:aws:iam::123456789012:role/temp-role" \
   --role-session-name "TempDirectoryUserRole" \
   --provided-contexts ProviderArn="arn:aws:iam::aws:contextProvider/
IdentityCenter",ContextAssertion="value from sts:identity_context"
```

You can now use the credentials received from this call to invoke the `s3:GetDataAccess` API operation and receive the final credentials with access to your S3 resources.

Getting started with S3 Access Grants

Amazon S3 Access Grants is an Amazon S3 feature that provides a scalable access control solution for your S3 data. S3 Access Grants is an S3 credential vendor, meaning that you register with S3 Access Grants your list of grants and at what level. Thereafter, when users or clients need access to your S3 data, they first ask S3 Access Grants for credentials. If there is a corresponding grant that authorizes access, S3 Access Grants vends temporary, least-privilege access credentials. The users or clients can then use S3 Access Grants vended credentials to access your S3 data. With that in mind, if your S3 data requirements mandate a complex or large permission configuration, you can use S3 Access Grants to scale S3 data permissions for users, groups, roles, and applications.

For most use cases, you can manage access control for your S3 data by using AWS Identity and Access Management (IAM) with bucket policies or IAM identity-based policies.

However, if you have complex S3 access control requirements, such as the following, you could benefit greatly from using S3 Access Grants:

- You are running into the bucket policy size limit of 20 KB.

- You grant human identities, for example, Microsoft Entra ID (formerly Azure Active Directory), Okta, or Ping users and groups, access to S3 data for analytics and big data.
- You must provide cross-account access without making frequent updates to IAM policies.
- Your data is unstructured and object-level rather than structured, in row and column format.

The S3 Access Grants workflow is as follows:

Steps	Description
1	Create an S3 Access Grants instance To get started, initiate an S3 Access Grants instance that will contain your individual access grants.
2	Register a location Second, register an S3 data location (such as the default, `s3://`) and then specify a default IAM role that S3 Access Grants assumes when providing access to the S3 data location. You can also add custom locations to specific buckets or prefixes and map those to custom IAM roles.
3	Create grants Create individual permission grants. Specify in these permissio n grants the registered S3 location, the scope of data access within the location, the identity of the grantee, and their access level (`READ`, `WRITE`, or `READWRITE`).
4	Request access to S3 data When users, applications, and AWS services want to access S3 data, they first make an access request. S3 Access Grants determines if the request should be authorized. If there is a corresponding grant that authorizes access, S3 Access Grants uses the registered location's IAM role that's associated with that grant to vend temporary credentials back to the request er.

Steps	Description
5	Access S3 data Applications use the temporary credentials vended by S3 Access Grants to access S3 data.

Working with S3 Access Grants instances

To get started with using AmazonS3 Access Grants, you first create an S3 Access Grants instance. You can create only one S3 Access Grants instance per AWS Region per account. The S3 Access Grants instance serves as the container for your S3 Access Grants resources, which include registered locations and grants.

With S3 Access Grants, you can create permission grants to your S3 data for AWS Identity and Access Management (IAM) users and roles. If you've added your corporate identity directory to AWS IAM Identity Center, you can associate this IAM Identity Center instance of your corporate directory with your S3 Access Grants instance. After you've done so, you can create access grants for your corporate users and groups. If you haven't yet added your corporate directory to IAM Identity Center, you can associate your S3 Access Grants instance with an IAM Identity Center instance later.

Topics

- Create an S3 Access Grants instance
- Get the details of an S3 Access Grants instance
- List your S3 Access Grants instances
- Associate or disassociate your IAM Identity Center instance
- Delete an S3 Access Grants instance

Create an S3 Access Grants instance

To get started with using AmazonS3 Access Grants, you first create an S3 Access Grants instance. You can create only one S3 Access Grants instance per AWS Region per account. The S3 Access Grants instance serves as the container for your S3 Access Grants resources, which include registered locations and grants.

With S3 Access Grants, you can create permission grants to your S3 data for AWS Identity and Access Management (IAM) users and roles. If you've added your corporate identity directory to AWS IAM Identity Center, you can associate this IAM Identity Center instance of your corporate directory with your S3 Access Grants instance. After you've done so, you can create access grants for your corporate users and groups. If you haven't yet added your corporate directory to IAM Identity Center, you can associate your S3 Access Grants instance with an IAM Identity Center instance later.

You can create an S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

Before you can grant access to your S3 data with S3 Access Grants, you must first create an S3 Access Grants instance in the same AWS Region as your S3 data.

Prerequisites

If you want to grant access to your S3 data by using identities from your corporate directory, add your corporate identity directory to AWS IAM Identity Center. If you're not yet ready to do so, you can associate your S3 Access Grants instance with an IAM Identity Center instance later.

To create an S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to switch to.
3. In the left navigation pane, choose **Access Grants**.
4. On the **S3 Access Grants** page, choose **Create S3 Access Grants instance**.
    a. In **Step 1** of the **Set up Access Grants instance** wizard, verify that you want to create the instance in the current AWS Region. Make sure that this is the same AWS Region where your S3 data is located. You can create one S3 Access Grants instance per AWS Region per account.
    b. (Optional) If you've added your corporate identity directory to AWS IAM Identity Center, you can associate this IAM Identity Center instance of your corporate directory with your S3 Access Grants instance.

To do so, select **Add IAM Identity Center instance in *region***. Then enter the IAM Identity Center instance Amazon Resource Name (ARN).

If you haven't yet added your corporate directory to IAM Identity Center, you can associate your S3 Access Grants instance with an IAM Identity Center instance later.

c. To create the S3 Access Grants instance, choose **Next**. To register a location, see Step 2 - register a location.

5. If **Next** or **Create S3 Access Grants instance** is disabled:

Cannot create instance

- You might already have an S3 Access Grants instance in the same AWS Region. In the left navigation pane, choose **Access Grants**. On the **S3 Access Grants** page, scroll down to the **S3 Access Grants instance in your account** section o determine if an instance already exists.
- You might not have the `s3:CreateAccessGrantsInstance` permission which is required to create an S3 Access Grants instance. Contact your account administrator. For additional permissions that are required if you are associating an IAM Identity Center instance, with your S3 Access Grants instance, see CreateAccessGrantsInstance .

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

Example Create an S3 Access Grants instance

```
aws s3control create-access-grants-instance \
--account-id 111122223333 \
--region us-east-2
```

Response:

```
{
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00",
    "AccessGrantsInstanceId": "default",
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
```

```
}
```

Using the REST API

You can use the Amazon S3 REST API to create an S3 Access Grants instance. For information on the REST API support for managing an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- AssociateAccessGrantsIdentityCenter
- CreateAccessGrantsInstance
- DeleteAccessGrantsInstance
- DissociateAccessGrantsIdentityCenter
- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix
- GetAccessGrantsInstanceResourcePolicy
- ListAccessGrantsInstances
- PutAccessGrantsInstanceResourcePolicy

Using the AWS SDKs

This section provides an example of how to create an S3 Access Grants instance by using the AWS SDKs.

Java

This example creates the S3 Access Grants instance, which serves as a container for your individual access grants. You can have one S3 Access Grants instance per AWS Region in your account. The response includes the instance ID `default` and an Amazon Resource Name (ARN) that's generated for your S3 Access Grants instance.

Example Create an S3 Access Grants instance request

```
public void createAccessGrantsInstance() {
CreateAccessGrantsInstanceRequest createRequest =
 CreateAccessGrantsInstanceRequest.builder().accountId("111122223333").build();
CreateAccessGrantsInstanceResponse createResponse =
 s3Control.createAccessGrantsInstance(createRequest);LOGGER.info("CreateAccessGrantsInstance
 " + createResponse);
```

```
}
```

Response:

```
CreateAccessGrantsInstanceResponse(
CreatedAt=2023-06-07T01:46:20.507Z,
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default)
```

Get the details of an S3 Access Grants instance

You can get the details of your Amazon S3 Access Grants instance in a particular AWS Region. You can get the details of your S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

Using the S3 console

To get the details of an S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. The **S3 Access Grants** page lists your S3 Access Grants instances and any cross-account instances that have been shared with your account. To view the details of an instance, choose **View details**.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

Example– Get the details of an S3 Access Grants instance

```
aws s3control get-access-grants-instance \
 --account-id 111122223333 \
```

```
 --region us-east-2
```

Response:

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
}
```

Using the REST API

For information about the Amazon S3 REST API support for managing an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- GetAccessGrantsInstance
- GetAccessGrantsInstanceForPrefix

Using the AWS SDKs

This section provides examples of how to get the details of an S3 Access Grants instance by using the AWS SDKs.

To use the following examples, replace the *`user input placeholders`* with your own information.

Java

Example– Get an S3 Access Grants instance

```
public void getAccessGrantsInstance() {
GetAccessGrantsInstanceRequest getRequest = GetAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
GetAccessGrantsInstanceResponse getResponse =
 s3Control.getAccessGrantsInstance(getRequest);
LOGGER.info("GetAccessGrantsInstanceResponse: " + getResponse);
}
```

Response:

```
GetAccessGrantsInstanceResponse(
AccessGrantsInstanceArn=arn:aws:s3:us-east-2: 111122223333:access-grants/default,
CreatedAt=2023-06-07T01:46:20.507Z)
```

List your S3 Access Grants instances

You can list your S3 Access Grants instances, including the instances that have been shared with you through AWS Resource Access Manager (AWS RAM).

You can list your S3 Access Grants instances by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

Using the S3 console

To list your S3 Access Grants instances

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. The **S3 Access Grants** page lists your S3 Access Grants instances and any cross-account instances that have been shared with your account. To view the details of an instance, choose **View details**.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

Example– List all S3 Access Grants instances for an account

This action lists the S3 Access Grants instances for an account. You can only have one S3 Access Grants instance per AWS Region. This action also lists other cross-account S3 Access Grants instances that your account has access to.

```
aws s3control list-access-grants-instances \
```

```
 --account-id 111122223333 \
 --region us-east-2
```

Response:

```
{
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default",
    "AccessGrantsInstanceId": "default",
    "CreatedAt": "2023-05-31T17:54:07.893000+00:00"
}
```

Using the REST API

For information about the Amazon S3 REST API support for managing an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- ListAccessGrantsInstances

Using the AWS SDKs

This section provides examples of how to get the details of an S3 Access Grants instance by using the AWS SDKs.

To use the following examples, replace the *`user input placeholders`* with your own information.

Java

Example– List all S3 Access Grants instances for an account

This action lists the S3 Access Grants instances for an account. You can only have one S3 Access Grants instance per Region. This action can also list other cross-account S3 Access Grants instances that your account has access to.

```
public void listAccessGrantsInstances() {
ListAccessGrantsInstancesRequest listRequest =
 ListAccessGrantsInstancesRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsInstancesResponse listResponse =
 s3Control.listAccessGrantsInstances(listRequest);
```

```
LOGGER.info("ListAccessGrantsInstancesResponse: " + listResponse);
}
```

Response:

```
ListAccessGrantsInstancesResponse(
AccessGrantsInstancesList=[
ListAccessGrantsInstanceEntry(
AccessGrantsInstanceId=default,
AccessGrantsInstanceArn=arn:aws:s3:us-east-2:111122223333:access-grants/default,
CreatedAt=2023-06-07T04:28:11.728Z
)
]
)
```

Associate or disassociate your IAM Identity Center instance

In Amazon S3 Access Grants, you can associate the AWS IAM Identity Center instance of your corporate identity directory with an S3 Access Grants instance. After you do so, you can create access grants for your corporate directory users and groups, in addition to AWS Identity and Access Management (IAM) users and roles.

If you no longer want to create access grants for your corporate directory users and groups, you can disassociate your IAM Identity Center instance from your S3 Access Grants instance.

You can associate or disassociate an IAM Identity Center instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

Using the S3 console

Before you associate your IAM Identity Center instance with your S3 Access Grants instance, you must add your corporate identity directory to IAM Identity Center. For more information, see the section called “S3 Access Grants and corporate directory identities”.

To associate an IAM Identity Center instance with an S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.

3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the details page, in the **IAM Identity Center** section, choose to either **Add** an IAM Identity Center instance or **Deregister** an already associated IAM Identity Center instance.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

Example– Associate an IAM Identity Center instance with an S3 Access Grants instance

```
aws s3control associate-access-grants-identity-center \
 --account-id 111122223333 \
 --identity-center-arn arn:aws:sso:::instance/ssoins-1234a567bb89012c \
 --profile access-grants-profile \
 --region eu-central-1

// No response body
```

Example– Disassociate an IAM Identity Center instance from an S3 Access Grants instance

```
aws s3control dissociate-access-grants-identity-center \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region eu-central-1

// No response body
```

Using the REST API

For information about the Amazon S3 REST API support for managing the association between an IAM Identity Center instance and an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- AssociateAccessGrantsIdentityCenter
- DissociateAccessGrantsIdentityCenter

Delete an S3 Access Grants instance

You can delete an Amazon S3 Access Grants instance from an AWS Region in your account. However, before you can delete an S3 Access Grants instance, you must first do the following:

- Delete all resources within the S3 Access Grants instance, including all grants and locations. For more information, see Delete a grant and Delete a location.
- If you've associated an AWS IAM Identity Center instance with your S3 Access Grants instance, you must disassociate the IAM Identity Center instance. For more information, see Associate or disassociate your IAM Identity Center instance.

> **Important**
>
> If you delete an S3 Access Grants instance, the deletion is permanent and can't be undone. All grantees that were given access through the grants in this S3 Access Grants instance will lose access to your S3 data.

You can delete an S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

Using the S3 console

To delete an S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the instance details page, choose **Delete instance**.
6. In the dialog box that appears, choose **Delete**. This action can't be undone.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *user input placeholders* with your own information.

> **Note**
>
> Before you can delete an S3 Access Grants instance, you must first delete all grants and locations created within the S3 Access Grants instance. If you have associated an IAM Identity Center center instance with your S3 Access Grants instance, you must disassociate it first.

Example– Delete an S3 Access Grants instance

```
aws s3control delete-access-grants-instance \
--account-id 111122223333 \
--profile access-grants-profile \
--region us-east-2 \
--endpoint-url https://s3-control.us-east-2.amazonaws.com \

 // No response body
```

Using the REST API

For information about the Amazon S3 REST API support for deleting an S3 Access Grants instance, see DeleteAccessGrantsInstance in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

This section provides examples of how to delete an S3 Access Grants instance by using the AWS SDKs.

To use the following example, replace the *user input placeholders* with your own information.

Java

> **Note**
>
> Before you can delete an S3 Access Grants instance, you must first delete all grants and locations created within the S3 Access Grants instance. If you have associated an

IAM Identity Center center instance with your S3 Access Grants instance, you must disassociate it first.

Example– Delete an S3 Access Grants instance

```
public void deleteAccessGrantsInstance() {
DeleteAccessGrantsInstanceRequest deleteRequest =
 DeleteAccessGrantsInstanceRequest.builder()
.accountId("111122223333")
.build();
DeleteAccessGrantsInstanceResponse deleteResponse =
 s3Control.deleteAccessGrantsInstance(deleteRequest);
LOGGER.info("DeleteAccessGrantsInstanceResponse: " + deleteResponse);
}
```

Working with S3 Access Grants locations

After you create an Amazon S3 Access Grants instance in an AWS Region in your account, you register an S3 location in that instance. An S3 Access Grants location maps the default S3 location (`s3://`), a bucket, or a prefix to an AWS Identity and Access Management (IAM) role. S3 Access Grants assumes this IAM role to vend temporary credentials to the grantee that is accessing that particular location. You must first register at least one location in your S3 Access Grants instance before you can create an access grant.

You can register a location, view a location's details, edit a location, and delete a location.

Note

After you register the first location in your S3 Access Grants instance, your instance still does not have any individual access grants in it. To create an access grant, see Create grants.

Topics

- Register a location
- View the details of a registered location

- Update a registered location
- Delete a registered location

Register a location

After you create an Amazon S3 Access Grants instance in an AWS Region in your account, you register an S3 location in that instance. An S3 Access Grants location maps the default S3 location (`s3://`), a bucket, or a prefix to an AWS Identity and Access Management (IAM) role. S3 Access Grants assumes this IAM role to vend temporary credentials to the grantee that is accessing that particular location. You must first register at least one location in your S3 Access Grants instance before you can create an access grant.

Recommended use case

We recommend that you register the default location (`s3://`) and map it to an IAM role. The location at the default S3 path (`s3://`) covers access to all of your S3 buckets in that AWS Region of your account. When you create an access grant, you can narrow the grant scope to a bucket, a prefix, or an object within the default location.

Complex access-management use cases

More complex access-management use cases might require you to register more than the default location. Some examples of such use cases are:

- Suppose that the *`amzn-s3-demo-bucket`* is a registered location in your S3 Access Grants instance with an IAM role mapped to it, but this IAM role is denied access to a particular prefix within the bucket. In this case, you can register the prefix that the IAM role does not have access to as a separate location and map that location to a different IAM role with the necessary access.
- Suppose that you want to create grants that restrict access to only the users within a virtual private cloud (VPC) endpoint. In this case, you can register a location for a bucket in which the IAM role restricts access to the VPC endpoint. Later, when a grantee asks S3 Access Grants for credentials, S3 Access Grants assumes the location's IAM role to vend the temporary credentials. This credential will deny access to the specific bucket unless the caller is within the VPC endpoint. This deny permission is applied in addition to the regular READ, WRITE, or READWRITE permission specified in the grant.

When you register a location, you must also specify the IAM role that S3 Access Grants assumes to vend temporary credentials and to scope the permissions for a specific grant.

If your use case requires you to register multiple locations in your S3 Access Grants instance, you can register any of the following:

S3 URI	IAM role	Description
`s3://`	*`Default-IAM-role`*	The default location, `s3://`, includes all buckets in the AWS Region.
`s3://`*`amzn-s3-demo-bucket1`*`/`	*`IAM-role-For-bucket`*	This location includes all objects in the specified bucket.
`s3://`*`amzn-s3-demo-bucket1`*`/`*`prefix-name`*	*`IAM-role-For-prefix`*	This location includes all objects in the bucket with an object key name that starts with this prefix.

Before you can register a specific bucket or prefix, make sure that you do the following:

- Create one or more buckets that contain the data that you want to grant access to. These buckets must be located in the same AWS Region as your S3 Access Grants instance. For more information, see Creating a bucket.

  Adding a prefix is an optional step. Prefixes are strings at the beginning of an object key name. You can use them to organize objects in your bucket as well as for access management. To add a prefix to a bucket, see Creating object key names.
- Create an IAM role that has permission to access your S3 data in the AWS Region. For more information, see Creating IAM roles in the *AWS IAM Identity Center user guide*.
- In the IAM role trust policy, give the S3 Access Grants service (`access-grants.s3.amazonaws.com`) principal access to the IAM role that you created. To do so, you can create a JSON file that contains the following statements. To add the trust policy to your account, see Create a role using custom trust policies.

  TestRolePolicy.json

  JSON

```
{
  "Version":"2012-10-17",
    "Statement": [
```

```
    {
      "Sid": "TestRolePolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "access-grants.s3.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "111122223333",
          "aws:SourceArn": "arn:aws:s3::111122223333:access-grants/default"
        }
      }
    }
  ]
}
```

Alternatively, for an IAM Identity Center use case, use the following policy which includes a second statement:

- Create an IAM policy to attach Amazon S3 permissions to the IAM role that you created. See the following example `iam-policy.json` file and replace the *`user input placeholders`* with your own information.

> **Note**
>
> - If you use server-side encryption with AWS Key Management Service (AWS KMS) keys to encrypt your data, the following example includes the necessary AWS KMS permissions for the IAM role in the policy. If you do not use this feature, you can remove these permissions from your IAM policy.
> - You can restrict the IAM role to access S3 data only if the credentials are vended by S3 Access Grants. This example shows you how to add a `Condition` statement for a specific S3 Access Grants instance. To use this `Condition`, replace the S3 Access Grants instance ARN in the `Condition` statement with your S3 Access Grants instance ARN. This has the format: `arn:aws:s3:`*`region`*`:`*`accountId`*`:access-grants/default`

iam-policy.json

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
         "Sid": "ObjectLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:region:accountId:access-
grants/default"]
            }
        }
      },
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect":"Allow",
         "Action":[
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:AbortMultipartUpload"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
```

```
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS
 Region:accountId:access-grants/default"]
            }
         }
      },
      {
         "Sid": "BucketLevelReadPermissions",
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::*"
         ],
         "Condition":{
            "StringEquals": { "aws:ResourceAccount": "accountId" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:AWS
 Region:accountId:access-grants/default"]
            }
         }
      },
   //Optionally add the following section if you use SSE-KMS encryption
      {
         "Sid": "KMSPermissions",
         "Effect":"Allow",
         "Action":[
            "kms:Decrypt",
            "kms:GenerateDataKey"
         ],
         "Resource":[
            "*"
         ]
      }
   ]
}
```

You can register a location in your S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, or the AWS SDKs.

> **Note**
>
> After you register the first location in your S3 Access Grants instance, your instance still does not have any individual access grants in it. To create an access grant, see Create grants.

Using the S3 console

Before you can grant access to your S3 data with S3 Access Grants, you must have at least one registered location.

To register a location in your S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.

   If you're using S3 Access Grants instance for the first time, make sure that you have completed Step 1 - create an S3 Access Grants instance and navigated to **Step 2** of the **Set up Access Grants instance** wizard. If you already have an S3 Access Grants instance, choose **View details**, and then from the **Locations** tab, choose **Register location**.

   a. For the **Location scope**, choose **Browse S3** or enter the S3 URI path to the location that you want to register. For S3 URI formats, see the location formats table. After you enter a URI, you can choose **View** to browse the location.

   b. For the **IAM role**, choose one of the following:

   - **Choose from existing IAM roles**

     Choose an IAM role from the dropdown list. After you choose a role, choose **View** to make sure that this role has the necessary permissions to manage the location that you're registering. Specifically, make sure that this role grants S3 Access Grants the permissions `sts:AssumeRole` and `sts:SetSourceIdentity`.

   - **Enter IAM role ARN**

Navigate to the IAM Console. Copy the Amazon Resource Name (ARN) of the IAM role and paste it in this box.

c. To finish, choose **Next** or **Register location**.

4. Troubleshooting:

Cannot register location

- The location might already be registered.

  You might not have the `s3:CreateAccessGrantsLocation` permission to register locations. Contact your account administrator.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

You can register the default location, `s3://`, or a custom location in your S3 Access Grants instance. Make sure that you first create an IAM role with principal access to the location, and then make sure that you grant S3 Access Grants permission to assume this role.

To use the following example commands, replace the *`user input placeholders`* with your own information.

Example Create a resource policy

Create a policy that allows S3 Access Grants to assume the IAM role. To do so, you can create a JSON file that contains the following statements. To add the resource policy to your account, see Create and attach your first customer managed policy.

TestRolePolicy.json

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1234567891011",
      "Action": ["sts:AssumeRole", "sts:SetSourceIdentity"],
```

```
      "Effect": "Allow",
      "Principal": {"Service": "access-grants.s3.amazonaws.com"}
    }
  ]
}
```

Example Create the role

Run the following IAM command to create the role.

```
aws iam create-role --role-name accessGrantsTestRole \
 --region us-east-2 \
 --assume-role-policy-document file://TestRolePolicy.json
```

Running the `create-role` command returns the policy:

```
{
    "Role": {
        "Path": "/",
        "RoleName": "accessGrantsTestRole",
        "RoleId": "AROASRDGX4WM4GH55GIDA",
        "Arn": "arn:aws:iam::111122223333:role/accessGrantsTestRole",
        "CreateDate": "2023-05-31T18:11:06+00:00",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Sid": "Stmt1685556427189",
                    "Action": [
                        "sts:AssumeRole",
                        "sts:SetSourceIdentity"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service":"access-grants.s3.amazonaws.com"
                    }
                }
            ]
        }
    }
}
```

Example

Create an IAM policy to attach Amazon S3 permissions to the IAM role. See the following example `iam-policy.json` file and replace the *`user input placeholders`* with your own information.

> **Note**
>
> If you use server-side encryption with AWS Key Management Service (AWS KMS) keys to encrypt your data, the following example adds the necessary AWS KMS permissions for the IAM role in the policy. If you do not use this feature, you can remove these permissions from your IAM policy.
> To make sure that the IAM role can only be used to access data in S3 if the credentials are vended out by S3 Access Grants, this example shows you how to add a `Condition` statement that specifies the S3 Access Grants instance (`s3:AccessGrantsInstance:`*`InstanceArn`*) in your IAM policy. When using following example policy, replace the *`user input placeholders`* with your own information.

iam-policy.json

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
         "Sid": "ObjectLevelReadPermissions",
         "Effect": "Allow",
         "Action": [
            "s3:GetObject",
            "s3:GetObjectVersion",
            "s3:GetObjectAcl",
            "s3:GetObjectVersionAcl",
            "s3:ListMultipartUploadParts"
         ],
         "Resource": [
            "arn:aws:s3:::*"
         ],
         "Condition": {
```

```
            "StringEquals": { "aws:ResourceAccount": "111122223333" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:us-east-1::access-
grants/default"]
            }
        }
      },
      {
         "Sid": "ObjectLevelWritePermissions",
         "Effect": "Allow",
         "Action": [
            "s3:PutObject",
            "s3:PutObjectAcl",
            "s3:PutObjectVersionAcl",
            "s3:DeleteObject",
            "s3:DeleteObjectVersion",
            "s3:AbortMultipartUpload"
         ],
         "Resource": [
            "arn:aws:s3:::*"
         ],
         "Condition": {
            "StringEquals": { "aws:ResourceAccount": "111122223333" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:us-east-1::access-
grants/default"]
            }
         }
      },
      {
         "Sid": "BucketLevelReadPermissions",
         "Effect": "Allow",
         "Action": [
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::*"
         ],
         "Condition": {
            "StringEquals": { "aws:ResourceAccount": "111122223333" },
            "ArnEquals": {
                "s3:AccessGrantsInstanceArn": ["arn:aws:s3:us-east-1::access-
grants/default"]
            }
```

```
            }
        },
        {
            "Sid": "KMSPermissions",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "StringEquals": {
                    "kms:ViaService": "s3.us-east-1.amazonaws.com"
                }
            }
        }
    ]
}
```

Example

Run the following command:

```
aws iam put-role-policy \
--role-name accessGrantsTestRole \
--policy-name accessGrantsTestRole \
--policy-document file://iam-policy.json
```

Example Register the default location

```
aws s3control create-access-grants-location \
 --account-id 111122223333 \
 --location-scope s3:// \
 --iam-role-arn arn:aws:iam::111122223333:role/accessGrantsTestRole
```

Response:

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
```

```
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://"
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

Example Register a custom location

```
aws s3control create-access-grants-location \
 --account-id 111122223333 \
 --location-scope s3://DOC-BUCKET-EXAMPLE/ \
 --iam-role-arn arn:aws:iam::123456789012:role/accessGrantsTestRole
```

Response:

```
{"CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2: 111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://DOC-BUCKET-EXAMPLE/",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

Using the REST API

For information about Amazon S3 REST API support for managing an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- CreateAccessGrantsLocation
- DeleteAccessGrantsLocation
- GetAccessGrantsLocation
- ListAccessGrantsLocations
- UpdateAccessGrantsLocation

Using the AWS SDKs

This section provides examples of how to register locations by using the AWS SDKs.

To use the following examples, replace the *user input placeholders* with your own information.

Java

You can register the default location, `s3://`, or a custom location in your S3 Access Grants instance. Make sure that you first create an IAM role with principal access to the location, and then make sure that you grant S3 Access Grants permission to assume this role.

To use the following example commands, replace the *user input placeholders* with your own information.

Example Register a default location

Request:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://")
.iamRoleArn("arn:aws:iam::123456789012:role/accessGrantsTestRole")
.build();
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

Response:

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:11.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

Example Register a custom location

Request:

```
public void createAccessGrantsLocation() {
CreateAccessGrantsLocationRequest createRequest =
 CreateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.locationScope("s3://DOC-BUCKET-EXAMPLE/")
.iamRoleArn("arn:aws:iam::111122223333:role/accessGrantsTestRole")
.build();
CreateAccessGrantsLocationResponse createResponse =
 s3Control.createAccessGrantsLocation(createRequest);
LOGGER.info("CreateAccessGrantsLocationResponse: " + createResponse);
}
```

Response:

```
CreateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=18cfe6fb-eb5a-4ac5-aba9-8d79f04c2012,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/18cfe6fb-eb5a-4ac5-aba9-8d79f04c2666,
LocationScope= s3://test-bucket-access-grants-user123/,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

View the details of a registered location

You can get the details of a location that's registered in your S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

Using the S3 console

To view the locations registered in your S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.

5. On the details page for the instance, choose the **Locations** tab.
6. Find the registered location that you want to view. To filter the list of registered locations, use the search box.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *user input placeholders* with your own information.

Example– Get the details of a registered location

```
aws s3control get-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id default
```

Response:

```
{
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
}
```

Example– List all of the locations that are registered in an S3 Access Grants instance

To restrict the results to an S3 prefix or bucket, you can optionally use the `--location-scope s3://bucket-and-or-prefix` parameter.

```
aws s3control list-access-grants-locations \
--account-id 111122223333 \
--region us-east-2
```

Response:

```
{"AccessGrantsLocationsList": [
```

```
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "default",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/default",
    "LocationScope": "s3://"
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     },
  {
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb456",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://amzn-s3-demo-bucket/prefixA*",
    "IAMRoleArn": "arn:aws:iam::111122223333:role/accessGrantsTestRole"
     }
   ]
  }
```

Using the REST API

For information about the Amazon S3 REST API support for getting the details of a registered location or listing all of the locations that are registered with an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- GetAccessGrantsLocation
- ListAccessGrantsLocations

Using the AWS SDKs

This section provides examples of how to get the details of a registered location or list all of the registered locations in an S3 Access Grants instance by using the AWS SDKs.

To use the following examples, replace the *`user input placeholders`* with your own information.

Java

Example– Get the details of a registered location

```
public void getAccessGrantsLocation() {
```

```
GetAccessGrantsLocationRequest getAccessGrantsLocationRequest =
 GetAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("default")
.build();
GetAccessGrantsLocationResponse getAccessGrantsLocationResponse =
 s3Control.getAccessGrantsLocation(getAccessGrantsLocationRequest);
LOGGER.info("GetAccessGrantsLocationResponse: " + getAccessGrantsLocationResponse);
}
```

Response:

```
GetAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope= s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
```

Example– List all registered locations in an S3 Access Grants instance

To restrict the results to an S3 prefix or bucket, you can optionally pass an S3 URI, such as `s3://bucket-and-or-prefix`, in the `LocationScope` parameter.

```
public void listAccessGrantsLocations() {

ListAccessGrantsLocationsRequest listRequest =
 ListAccessGrantsLocationsRequest.builder()
.accountId("111122223333")
.build();

ListAccessGrantsLocationsResponse listResponse =
 s3Control.listAccessGrantsLocations(listRequest);
LOGGER.info("ListAccessGrantsLocationsResponse: " + listResponse);
}
```

Response:

```
ListAccessGrantsLocationsResponse(
AccessGrantsLocationsList=[
```

```
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:11.027Z,
AccessGrantsLocationId=default,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/default,
LocationScope=s3://,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
),
ListAccessGrantsLocationsEntry(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb456,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://amzn-s3-demo-bucket/prefixA*,
IAMRoleArn=arn:aws:iam::111122223333:role/accessGrantsTestRole
)
]
)
```

Update a registered location

You can update the AWS Identity and Access Management (IAM) role of a location that's registered in your Amazon S3 Access Grants instance. For each new IAM role that you use to register a location in S3 Access Grants, be sure to give the S3 Access Grants service principal (`access-grants.s3.amazonaws.com`) access to this role. To do this, add an entry for the new IAM role in the same trust policy JSON file that you used when you first registered the location.

You can update a location in your S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

Using the S3 console

To update the IAM role of a location registered with your S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.

5. On the details page for the instance, choose the **Locations** tab.
6. Find the location that you want to update. To filter the list of locations, use the search box.
7. Choose the options button next to the registered location that you want to update.
8. Update the IAM role, and then choose **Save changes**.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

Example– Update the IAM role of a registered location

```
aws s3control update-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id 635f1139-1af2-4e43-8131-a4de006eb999 \
--iam-role-arn arn:aws:iam::777788889999:role/accessGrantsTestRole
```

Response:

```
{
    "CreatedAt": "2023-05-31T18:23:48.107000+00:00",
    "AccessGrantsLocationId": "635f1139-1af2-4e43-8131-a4de006eb999",
    "AccessGrantsLocationArn": "arn:aws:s3:us-east-2:777788889999:access-grants/
default/location/635f1139-1af2-4e43-8131-a4de006eb888",
    "LocationScope": "s3://amzn-s3-demo-bucket/prefixB*",
    "IAMRoleArn": "arn:aws:iam::777788889999:role/accessGrantsTestRole"
}
```

Using the REST API

For information on the Amazon S3 REST API support for updating a location in an S3 Access Grants instance, see UpdateAccessGrantsLocation in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

This section provides examples of how to update the IAM role of a registered location by using the AWS SDKs.

To use the following example, replace the *user input placeholders* with your own information.

Java

Example– Update the IAM role of a registered location

```
public void updateAccessGrantsLocation() {
UpdateAccessGrantsLocationRequest updateRequest =
 UpdateAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("635f1139-1af2-4e43-8131-a4de006eb999")
.iamRoleArn("arn:aws:iam::777788889999:role/accessGrantsTestRole")
.build();
UpdateAccessGrantsLocationResponse updateResponse =
 s3Control.updateAccessGrantsLocation(updateRequest);
LOGGER.info("UpdateAccessGrantsLocationResponse: " + updateResponse);
}
```

Response:

```
UpdateAccessGrantsLocationResponse(
CreatedAt=2023-06-07T04:35:10.027Z,
AccessGrantsLocationId=635f1139-1af2-4e43-8131-a4de006eb999,
AccessGrantsLocationArn=arn:aws:s3:us-east-2:777788889999:access-grants/default/
location/635f1139-1af2-4e43-8131-a4de006eb888,
LocationScope=s3://amzn-s3-demo-bucket/prefixB*,
IAMRoleArn=arn:aws:iam::777788889999:role/accessGrantsTestRole
)
```

Delete a registered location

You can delete a location registration from an Amazon S3 Access Grants instance. Deleting the location deregisters it from the S3 Access Grants instance.

Before you can remove a location registration from an S3 Access Grants instance, you must delete all of the grants that are associated with this location. For information about how to delete grants, see Delete a grant.

You can delete a location in your S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

Using the S3 console

To delete a location registration from your S3 Access Grants instance

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the details page for the instance, choose the **Locations** tab.
6. Find the location that you want to update. To filter the list of locations, use the search box.
7. Choose the option button next to the registered location that you want to delete.
8. Choose **Deregister**.
9. A dialog box appears that warns you that this action can't be undone. To delete the location, choose **Deregister**.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

Example– Delete a location registration

```
aws s3control delete-access-grants-location \
--account-id 111122223333 \
--access-grants-location-id  a1b2c3d4-5678-90ab-cdef-EXAMPLE11111
 // No response body
```

Using the REST API

For information about the Amazon S3 REST API support for deleting a location from an S3 Access Grants instance, see DeleteAccessGrantsLocation in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

This section provides an example of how to delete a location by using the AWS SDKs.

To use the following example, replace the *user input placeholders* with your own information.

Java

Example– Delete a location registration

```
public void deleteAccessGrantsLocation() {
DeleteAccessGrantsLocationRequest deleteRequest =
 DeleteAccessGrantsLocationRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantsLocationResponse deleteResponse =
 s3Control.deleteAccessGrantsLocation(deleteRequest);
LOGGER.info("DeleteAccessGrantsLocationResponse: " + deleteResponse);
}
```

Response:

```
DeleteAccessGrantsLocationResponse()
```

Working with grants in S3 Access Grants

An individual access *grant* in an S3 Access Grants instance allows a specific identity—an AWS Identity and Access Management (IAM) principal, or a user or group in a corporate directory—to get access within a location that is registered in your S3 Access Grants instance. A location maps buckets or prefixes to an IAM role. S3 Access Grants assumes this IAM role to vend temporary credentials to grantees.

After you register at least one location in your S3 Access Grants instance, you can create an access grant.

The grantee can be an IAM user or role or a directory user or group. A directory user is a user from your corporate directory or external identity source that you associated with your S3 Access Grants instance. For more information, see S3 Access Grants and corporate directory

identities. To create a grant for a specific directory user or group from IAM Identity Center, find the GUID that IAM Identity Center uses to identify that user in IAM Identity Center, for example, `a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`. For more information about how to use IAM Identity Center to view user information, see View user and group assignments in the *AWS IAM Identity Center user guide*.

You can grant access to a bucket, a prefix, or an object. A prefix in Amazon S3 is a string of characters in the beginning of an object key name that is used to organize objects within a bucket. This can be any string of allowed characters, for example, object key names in your bucket that start with the `engineering/` prefix.

Topics

- Create grants
- View a grant
- Delete a grant

Create grants

An individual access *grant* in an S3 Access Grants instance allows a specific identity—an AWS Identity and Access Management (IAM) principal, or a user or group in a corporate directory—to get access within a location that is registered in your S3 Access Grants instance. A location maps buckets or prefixes to an IAM role. S3 Access Grants assumes this IAM role to vend temporary credentials to grantees.

After you register at least one location in your S3 Access Grants instance, you can create an access grant.

The grantee can be an IAM user or role or a directory user or group. A directory user is a user from your corporate directory or external identity source that you associated with your S3 Access Grants instance. For more information, see S3 Access Grants and corporate directory identities. To create a grant for a specific directory user or group from IAM Identity Center, find the GUID that IAM Identity Center uses to identify that user in IAM Identity Center, for example, `a1b2c3d4-5678-90ab-cdef-EXAMPLE11111`. For more information about how to use IAM Identity Center to view user information, see View user and group assignments in the *AWS IAM Identity Center user guide*.

You can grant access to a bucket, a prefix, or an object. A prefix in Amazon S3 is a string of characters in the beginning of an object key name that is used to organize objects within a bucket.

This can be any string of allowed characters, for example, object key names in your bucket that start with the `engineering/` prefix.

Subprefix

When granting access to a registered location, you can use the `Subprefix` field to narrow the scope of access to a subset of the location scope. If the registered location that you choose for the grant is the default S3 path (`s3://`), you must narrow the grant scope. You cannot create an access grant for the default location (`s3://`), which would give the grantee access to every bucket in an AWS Region. Instead, you must narrow the grant scope to one of the following:

- A bucket: `s3://`*`bucket`*`/*`
- A prefix within a bucket: `s3://`*`bucket`*`/`*`prefix`*`*`
- A prefix within a prefix: `s3://`*`bucket`*`/`*`prefixA`*`/`*`prefixB`*`*`
- An object: `s3://`*`bucket`*`/`*`object-key-name`*

If you are creating an access grant where the registered location is a bucket, you can pass one of the following in the `Subprefix` field to narrow the grant scope:

- A prefix within the bucket: *`prefix`*`*`
- A prefix within a prefix: *`prefixA/prefixB`*`*`
- An object: */`*`object-key-name`*

After you create the grant, the grant scope that's displayed in the Amazon S3 console or the `GrantScope` that is returned in the API or AWS Command Line Interface (AWS CLI) response is the result of concatenating the location path with the `Subprefix`. Make sure that this concatenated path maps correctly to the S3 bucket, prefix, or object to which you want to grant access.

Note

- If you need to create an access grant that grants access to only one object, you must specify that the grant type is for an object. To do this in an API call or a CLI command, pass the `s3PrefixType` parameter with the value `Object`. In the Amazon S3 console, when you create the grant, after you select a location, under **Grant Scope**, select the **Grant scope is an object** checkbox.

- You cannot create a grant to a bucket if the bucket does not yet exist. However, you can create a grant to a prefix that does not yet exist.
- For the maximum number of grants that you can create in your S3 Access Grants instance, see S3 Access Grants limitations.

You can create an access grant by using the Amazon S3 console, AWS CLI, the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To create an access grant

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.

   If you're using the S3 Access Grants instance for the first time, make sure that you have completed Step 2 - register a location and navigated to **Step 3** of the **Set up Access Grants instance** wizard. If you already have an S3 Access Grants instance, choose **View details**, and then from the **Grants** tab, choose **Create grant**.

   a. In the **Grant scope** section, select or enter a registered location.

      If you selected the default `s3://` location, use the **Subprefix** box to can narrow the scope of the access grant. For more information, see Subprefix. If you're granting access only to an object, select **Grant scope is an object**.

   b. Under **Permissions and access**, select the **Permission** level, either **Read**, **Write**, or both.

      Then choose the **Grantee type**. If you have added your corporate directory to IAM Identity Center and associated this IAM Identity Center instance with your S3 Access Grants instance, you can choose **Directory identity from IAM Identity Center**. If you choose this option, get the ID of the user or group from IAM Identity Center and enter it in this section.

If the **Grantee type** is an IAM user or role, choose **IAM principal**. Under **IAM principal type**, choose **User** or **Role**. Then, under **IAM principal user**, either choose from the list or enter the identity's ID.

c. To create the S3 Access Grants grant, choose **Next** or **Create grant**.

4. If **Next** or **Create grant** is disabled:

Cannot create grant

- You might need to register a location first in your S3 Access Grants instance.
- You might not have the `s3:CreateAccessGrant` permission to create an access grant. Contact your account administrator.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following examples show how to create an access grant request for an IAM principal and how to create an access grant request for a corporate directory user or group.

To use the following example commands, replace the *`user input placeholders`* with your own information.

> **Note**
>
> If you're creating an access grant that grants access to only one object, include the required parameter `--s3-prefix-type Object`.

Example Create an access grant request for an IAM principal

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222 \
--access-grants-location-configuration S3SubPrefix=prefixB* \
--permission READ \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::123456789012:user/data-consumer-3
```

Example Create an access grant response

```
{"CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "AccessGrantsLocationId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://amzn-s3-demo-bucket/prefix*",
    "Permission": "READ"
}
```

Create an access grant request for a directory user or group

To create an access grant request for a directory user or group, you must first get the GUID for the directory user or group by running one of the following commands.

Example Get a GUID for a directory user or group

You can find the GUID of an IAM Identity Center user through the IAM Identity Center console or by using the AWS CLI or AWS SDKs. The following command lists the users in he specified IAM Identity Center instance, with their names and identifiers.

```
aws identitystore list-users --identity-store-id d-1a2b3c4d1234
```

This command lists the groups in the specified IAM Identity Center instance.

```
aws identitystore list-groups --identity-store-id d-1a2b3c4d1234
```

Example Create an access grant for a directory user or group

This command is similar to creating a grant for IAM users or roles, except the grantee type is `DIRECTORY_USER` or `DIRECTORY_GROUP`, and the grantee identifier is the GUID for the directory user or group.

```
aws s3control create-access-grant \
--account-id 123456789012 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix="amzn-s3-demo-bucket/rafael/*" \
--permission READWRITE \
--grantee GranteeType=DIRECTORY_USER,GranteeIdentifier=83d43802-00b1-7054-db02-
f1d683aacba5 \
```

Using the REST API

For information about the Amazon S3 REST API support for managing access grants, see the following sections in the *Amazon Simple Storage Service API Reference*:

- CreateAccessGrant
- DeleteAccessGrant
- GetAccessGrant
- ListAccessGrants

Using the AWS SDKs

This section provides examples of how to create an access grant by using the AWS SDKs.

Java

To use the following example, replace the *`user input placeholders`* with your own information:

> **Note**
>
> If you are creating an access grant that grants access to only one object, include the required parameter `.s3PrefixType(S3PrefixType.Object)`.

Example Create an access grant request

```
public void createAccessGrant() {
CreateAccessGrantRequest createRequest = CreateAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantsLocationId("a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa")
```

```
.permission("READ")
.accessGrantsLocationConfiguration(AccessGrantsLocationConfiguration.builder().s3SubPrefix("
.grantee(Grantee.builder().granteeType("IAM").granteeIdentifier("arn:aws:iam::111122223333:u
data-consumer-3").build())
.build();
CreateAccessGrantResponse createResponse =
 s3Control.createAccessGrant(createRequest);
LOGGER.info("CreateAccessGrantResponse: " + createResponse);
}
```

Example Create an access grant response

```
CreateAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
AccessGrantArn=arn:aws:s3:us-east-2:444455556666:access-grants/default/grant/
a1b2c3d4-5678-90ab-cdef-EXAMPLE33333,
Grantee=Grantee(
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
),
AccessGrantsLocationId=a1b2c3d4-5678-90ab-cdef-EXAMPLEaaaaa,
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
S3SubPrefix=prefixB*
),
GrantScope=s3://amzn-s3-demo-bucket/prefixB,
Permission=READ
)
```

View a grant

You can view the details of an access grant in your Amazon S3 Access Grants instance by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

Using the S3 console

To view the details of an access grant

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the details page, choose the **Grants** tab.
6. In the **Grants** section, find the access grant that you want to view. To filter the list of grants, use the search box.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example commands, replace the *`user input placeholders`* with your own information.

Example– Get the details of an access grant

```
aws s3control get-access-grant \
--account-id 111122223333 \
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE22222
```

Response:

```
{
    "CreatedAt": "2023-05-31T18:41:34.663000+00:00",
    "AccessGrantId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
    "Grantee": {
        "GranteeType": "IAM",
        "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
    },
    "Permission": "READ",
    "AccessGrantsLocationId": "12a6710f-5af8-41f5-b035-0bc795bf1a2b",
    "AccessGrantsLocationConfiguration": {
        "S3SubPrefix": "prefixB*"
    },
    "GrantScope": "s3://amzn-s3-demo-bucket/"
}
```

Example– List all of the access grants in an S3 Access Grants instance

You can optionally use the following parameters to restrict the results to an S3 prefix or AWS Identity and Access Management (IAM) identity:

- **Subprefix** – `--grant-scope s3://bucket-name/prefix*`
- **IAM identity** – `--grantee-type IAM` and `--grantee-identifier arn:aws:iam::123456789000:role/accessGrantsConsumerRole`

```
aws s3control list-access-grants \
--account-id 111122223333
```

Response:

```
{
    "AccessGrantsList": [{"CreatedAt": "2023-06-14T17:54:46.542000+00:00",
            "AccessGrantId": "dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-3"
            },
            "Permission": "READ",
            "AccessGrantsLocationId": "23514a34-ea2e-4ddf-b425-d0d4bfcarda1",
            "GrantScope": "s3://amzn-s3-demo-bucket/prefixA*"
        },
        {"CreatedAt": "2023-06-24T17:54:46.542000+00:00",
            "AccessGrantId": "ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "AccessGrantArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2",
            "Grantee": {
                "GranteeType": "IAM",
                "GranteeIdentifier": "arn:aws:iam::111122223333:user/data-consumer-9"
            },
            "Permission": "READ",
            "AccessGrantsLocationId": "12414a34-ea2e-4ddf-b425-d0d4bfcacao0",
            "GrantScope": "s3://amzn-s3-demo-bucket/prefixB*"
        },

    ]
```

```
}
```

Using the REST API

You can use Amazon S3 API operations to view the details of an access grant and list all access grants in an S3 Access Grants instance. For information about the REST API support for managing access grants, see the following sections in the *Amazon Simple Storage Service API Reference*:

- GetAccessGrant
- ListAccessGrants

Using the AWS SDKs

This section provides examples of how to get the details of an access grant by using the AWS SDKs.

To use the following examples, replace the *`user input placeholders`* with your own information.

Java

Example– Get the details of an access grant

```
public void getAccessGrant() {
GetAccessGrantRequest getRequest = GetAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE22222")
.build();
GetAccessGrantResponse getResponse = s3Control.getAccessGrant(getRequest);
LOGGER.info("GetAccessGrantResponse: " + getResponse);
}
```

Response:

```
GetAccessGrantResponse(
CreatedAt=2023-06-07T05:20:26.330Z,
AccessGrantId=a1b2c3d4-5678-90ab-cdef-EXAMPLE22222,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant-fd3a5086-42f7-4b34-9fad-472e2942c70e,
Grantee=Grantee(
```

```
GranteeType=IAM,
GranteeIdentifier=arn:aws:iam::111122223333:user/data-consumer-3
),
Permission=READ,
AccessGrantsLocationId=12a6710f-5af8-41f5-b035-0bc795bf1a2b,
AccessGrantsLocationConfiguration=AccessGrantsLocationConfiguration(
S3SubPrefix=prefixB*
),
GrantScope=s3://amzn-s3-demo-bucket/
)
```

Example– List all of the access grants in an S3 Access Grants instance

You can optionally use these parameters to restrict the results to an S3 prefix or IAM identity:

- **Scope** – `GrantScope=s3://bucket-name/prefix*`
- **Grantee** – `GranteeType=IAM` and `GranteeIdentifier=arn:aws:iam::111122223333:role/accessGrantsConsumerRole`

```
public void listAccessGrants() {
ListAccessGrantsRequest listRequest = ListAccessGrantsRequest.builder()
.accountId("111122223333")
.build();
ListAccessGrantsResponse listResponse = s3Control.listAccessGrants(listRequest);
LOGGER.info("ListAccessGrantsResponse: " + listResponse);
}
```

Response:

```
ListAccessGrantsResponse(
AccessGrantsList=[
ListAccessGrantEntry(
CreatedAt=2023-06-14T17:54:46.540z,
AccessGrantId=dd8dd089-b224-4d82-95f6-975b4185bbaa,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/dd8dd089-b224-4d82-95f6-975b4185bbaa,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-3
),
```

```
Permission=READ,
AccessGrantsLocationId=23514a34-ea2e-4ddf-b425-d0d4bfcarda1,
GrantScope=s3://amzn-s3-demo-bucket/prefixA
),
ListAccessGrantEntry(
CreatedAt=2023-06-24T17:54:46.540Z,
AccessGrantId=ee8ee089-b224-4d72-85f6-975b4185a1b2,
AccessGrantArn=arn:aws:s3:us-east-2:111122223333:access-grants/default/
grant/ee8ee089-b224-4d72-85f6-975b4185a1b2,
Grantee=Grantee(
GranteeType=IAM, GranteeIdentifier= arn:aws:iam::111122223333:user/data-consumer-9
),
Permission=READ,
AccessGrantsLocationId=12414a34-ea2e-4ddf-b425-d0d4bfcacao0,
GrantScope=s3://amzn-s3-demo-bucket/prefixB*
)
]
)
```

Delete a grant

You can delete access grants from your Amazon S3 Access Grants instance. You can't undo an access grant deletion. After you delete an access grant, the grantee will no longer have access to your Amazon S3 data.

You can delete an access grant by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

Using the S3 console

To delete an access grant

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Access Grants**.
3. On the **S3 Access Grants** page, choose the Region that contains the S3 Access Grants instance that you want to work with.
4. Choose **View details** for the instance.
5. On the details page, choose the **Grants** tab.

6. Search for the grant that you want to delete. When you locate the grant, choose the radio button next to it.
7. Choose **Delete**. A dialog box appears with a warning that your action can't be undone. Choose **Delete** again to delete the grant.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

Example– Delete an access grant

```
aws s3control delete-access-grant \
--account-id 111122223333 \
--access-grant-id a1b2c3d4-5678-90ab-cdef-EXAMPLE11111

// No response body
```

Using the REST API

For information about the Amazon S3 REST API support for managing access grants, see DeleteAccessGrant in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

This section provides examples of how to delete an access grant by using the AWS SDKs. To use the following example, replace the *`user input placeholders`* with your own information.

Java

Example– Delete an access grant

```
public void deleteAccessGrant() {
DeleteAccessGrantRequest deleteRequest = DeleteAccessGrantRequest.builder()
.accountId("111122223333")
.accessGrantId("a1b2c3d4-5678-90ab-cdef-EXAMPLE11111")
.build();
DeleteAccessGrantResponse deleteResponse =
 s3Control.deleteAccessGrant(deleteRequest);
```

```
LOGGER.info("DeleteAccessGrantResponse: " + deleteResponse);
}
```

Response:

```
DeleteAccessGrantResponse()
```

Getting S3 data using access grants

Grantees who have been given access to S3 data through S3 Access Grants must request temporary credentials from S3 Access Grants, which they use to access the S3 data. For more information, see Request access to Amazon S3 data through S3 Access Grants. Grantees then use the temporary credentials to perform allowable S3 actions on the S3 data. For more information, see Accessing S3 data using credentials vended by S3 Access Grants. Grantees can optionally request a list of their access grants for an AWS account before requesting these credentials. For more information, see List the caller's access grants.

Topics

- Request access to Amazon S3 data through S3 Access Grants
- Accessing S3 data using credentials vended by S3 Access Grants
- List the caller's access grants

Request access to Amazon S3 data through S3 Access Grants

After you create an access grant using S3 Access Grants, grantees can request credentials to access the S3 data that they were granted access to. Grantees can be AWS Identity and Access Management (IAM) principals, your corporate directory identities, or authorized applications.

An application or AWS service can use the S3 Access Grants `GetDataAccess` API operation to ask S3 Access Grants for access to your S3 data on behalf of a grantee. `GetDataAccess` first verifies that you have granted this identity access to the data. Then, S3 Access Grants uses the AssumeRole API operation to obtain a temporary credential token and vends it to the requester. This temporary credential token is an AWS Security Token Service (AWS STS) token.

The `GetDataAccess` request must include the `target` parameter, which specifies the scope of the S3 data that the temporary credentials apply to. This `target` scope can be the same as the scope of the grant or a subset of that scope, but the `target` scope must be within the scope of the

grant that was given to the grantee. The request must also specify the `permission` parameter to indicate the permission level for the temporary credentials, whether `READ`, `WRITE`, or `READWRITE`.

Privilege

The requester can specify the privilege level of the temporary token in their credential request. Using the `privilege` parameter, the requester can reduce or increase the temporary credentials' scope of access, within the boundaries of the grant scope. The default value of the `privilege` parameter is `Default`, which means that the target scope of the credential returned is the original grant scope. The other possible value for `privilege` is `Minimal`. If the `target` scope is reduced from the original grant scope, then the temporary credential is de-scoped to match the `target` scope, as long as the `target` scope is within the grant scope.

The following table details the effect of the `privilege` parameter on two grants. One grant has the scope `S3://amzn-s3-demo-bucket1/bob/*`, which includes the entire bob/ prefix in the *amzn-s3-demo-bucket1* bucket. The other grant has the scope `S3://amzn-s3-demo-bucket1/bob/reports/*`, which includes only the bob/reports/ prefix in the *amzn-s3-demo-bucket1* bucket.

Grant scope	Requested scope	Privilege	Returned scope	Effect
S3://*amzn-s3-demo-bucket1* /bob/*	*amzn-s3-demo-bucket1* /bob/*	Default	*amzn-s3-demo-bucket1* /bob/*	The requester has access to all objects that have key names that start with the prefix *bob/* in the *amzn-s3-demo-bucket1* bucket.
S3://*amzn-s3-demo-bucket1* /bob/*	*amzn-s3-demo-bucket1* /bob/	Minimal	*amzn-s3-demo-bucket1* /bob/	Without a wild card * character after the prefix name bob/, the requester has access to only the object named bob/ in the *amzn-s3-demo-bucket1* bucket. It's not common to have such an object.

Grant scope	Requested scope	Privilege	Returned scope	Effect
				The requester doesn't have access to any other objects, including those that have key names that start with the bob/ prefix.
S3://*amzn-s3-demo-bucket1* /bob/*	*amzn-s3-demo-bucket1* /bob/images/*	Minimal	*amzn-s3-demo-bucket1* /bob/images/*	The requester has access to all objects that have key names that start with the prefix *bob/images/** in the *amzn-s3-demo-bucket1* bucket.
S3://*amzn-s3-demo-bucket1* /bob/reports/*	*amzn-s3-demo-bucket1* /bob/reports/file.txt	Default	*amzn-s3-demo-bucket1* /bob/reports/*	The requester has access to all objects that have key names that start with the bob/reports prefix in the *amzn-s3-demo-bucket1* bucket, which is the scope of the matching grant.
S3://*amzn-s3-demo-bucket1* /bob/reports/*	*amzn-s3-demo-bucket1* /bob/reports/file.txt	Minimal	*amzn-s3-demo-bucket1* /bob/reports/file.txt	The requester has access only to the object with the key name bob/reports/file.txt in the *amzn-s3-demo-bucket1* bucket. The requester has no access to any other object.

Directory identities

`GetDataAccess` considers all of the identities involved in a request when matching suitable grants. For corporate directory identities, `GetDataAccess` also returns the grants of the IAM identity that is used for the identity-aware session. For more information on identity-aware sessions, see Granting permissions to use identity-aware console sessions in the *AWS Identity and Access Management User Guide*. `GetDataAccess` generates credentials restricting scope to the most restrictive grant, as shown in the following table:

Grant scope for IAM identity	Grant scope for directory identity	Requested scope	Returned scope	Privilege	Effect
S3://*amzn-s3-demo-bucket1* / bob/*	*amzn-s3-demo-bucket1* / bob/images/*	S3://*amzn-s3-demo-bucket1* / bob/images/image1.jpeg	S3://*amzn-s3-demo-bucket1* / bob/images/*	Default	The requestor has access to all of the objects that have key names that start with the prefix *bob/* as a part of the grant for the IAM role but is restricted to the prefixes *bob/images/* as a part of the grant for the directory identity. Both the IAM role and directory identity provide access to the requested scope, which is bob/images/image1.jpeg , but the directory identity has a more restrictive grant. So, the returned scope is restricted to the more

Grant scope for IAM identity	Grant scope for directory identity	Requested scope	Returned scope	Privilege	Effect
					restrictive grant for the directory identity.
S3://amzn-s3-demo-bucket1 / bob/*	amzn-s3-demo-bucket1 / bob/images/*	S3://amzn-s3-demo-bucket1 / bob/images/image1.jpeg	S3://amzn-s3-demo-bucket1 / bob/images/image1.jpeg	Minimal	Because the Privilege is set to Minimal, even though the identity has access to a bigger scope, only the requested scope is returned bob/images/image1.jpeg .

Grant scope for IAM identity	Grant scope for directory identity	Requested scope	Returned scope	Privilege	Effect
S3://*amzn-s3-demo-bucket1* / bob/images/*	*amzn-s3-demo-bucket1* / bob/*	S3://*amzn-s3-demo-bucket1* / bob/images/image1.jpeg	S3://*amzn-s3-demo-bucket1* / bob/images/*	Default	The requestor has access to all of the objects that have key names that start with the prefix *bob/* as a part of the grant for the directory identity but is restricted to the prefixes *bob/images/* as a part of the grant for the IAM role. Both the IAM role and directory identity provide access to the requested scope, which is bob/images/image1.jpeg , but the IAM role has a more restrictive grant. So, the returned scope is restricted to the more restrictive grant for the IAM role.

Grant scope for IAM identity	Grant scope for directory identity	Requested scope	Returned scope	Privilege	Effect
S3://*amzn-s3-demo-bucket1* / bob/images/*	*amzn-s3-demo-bucket1* / bob/*	S3://*amzn-s3-demo-bucket1* / bob/images/image1.jpeg	S3://*amzn-s3-demo-bucket1* / bob/images/image1.jpeg	Minimal	Because the Privilege is set to Minimal, even though the identity has access to a bigger scope, only the requested scope is returned bob/images/image1.jpeg .

Duration

The durationSeconds parameter sets the temporary credential's duration, in seconds. The default value is 3600 seconds (1 hour), but the requester (the grantee) can specify a range from 900 seconds (15 minutes) up to 43200 seconds (12 hours). If the grantee requests a value higher than this maximum, the request fails.

Instance scope

The instanceScope parameter sets the scope of S3 Access Grants instances that S3 Access Grants searches for a grant that matches the target parameter. The default value is SELF, but the requester can set instanceScope to ALL. With SELF, the AWS account ID you pass in the request must be the account that owns the data, and S3 Access Grants returns credentials for the matching grant in that account's instance. If instances from other accounts are shared with you through AWS Resource Access Manager (AWS RAM), you can set instanceScope to ALL. S3 Access Grants then finds the account that owns the bucket in target and returns credentials for the matching grant in that account's instance. With ALL, the account ID in the request can be your own account ID. In both cases, the response includes the MatchedInstanceArn field, which identifies the instance that contains the matching grant. With SELF, this is the instance of the account ID in the request. With ALL, it's the instance where the grant matched, which can belong to another account.

> **Note**
>
> In your request for a temporary token, if the location is an object, set the value of the `targetType` parameter in your request to `Object`. This parameter is required only if the location is an object and the privilege level is `Minimal`. If the location is a bucket or a prefix, you don't need to specify this parameter.

Examples

You can request temporary credentials by using the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs. See these examples.

For additional information, see GetDataAccess in the *Amazon Simple Storage Service API Reference*.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

Example Request temporary credentials

Request:

```
aws s3control get-data-access \
--account-id 111122223333 \
--target s3://amzn-s3-demo-bucket/prefixA* \
--permission READ \
--privilege Default \
--region us-east-2
```

Response:

```
{
"Credentials": {
"AccessKeyId": "Example-key-id",
"SecretAccessKey": "Example-access-key",
"SessionToken": "Example-session-token",
"Expiration": "2023-06-14T18:56:45+00:00"},
"MatchedGrantTarget": "s3://amzn-s3-demo-bucket/prefixA**",
```

```
"Grantee": {
    "GranteeType": "IAM",
    "GranteeIdentifier": "arn:aws:iam::111122223333:role/role-name"
 }
}
```

Using the REST API

For information about the Amazon S3 REST API support for requesting temporary credentials from S3 Access Grants, see GetDataAccess in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

This section provides an example of how grantees request temporary credentials from S3 Access Grants by using the AWS SDKs.

Java

The following code example returns the temporary credentials that the grantee uses to access your S3 data. To use this code example, replace the *`user input placeholders`* with your own information.

Example Get temporary credentials

Request:

```
public void getDataAccess() {
GetDataAccessRequest getDataAccessRequest = GetDataAccessRequest.builder()
.accountId("111122223333")
.permission(Permission.READ)
.privilege(Privilege.MINIMAL)
.target("s3://amzn-s3-demo-bucket/prefixA*")
.build();
GetDataAccessResponse getDataAccessResponse =
 s3Control.getDataAccess(getDataAccessRequest);
LOGGER.info("GetDataAccessResponse: " + getDataAccessResponse);
}
```

Response:

```
GetDataAccessResponse(
Credentials=Credentials(
AccessKeyId="Example-access-key-id",
```

```
SecretAccessKey="Example-secret-access-key",
SessionToken="Example-session-token",
Expiration=2023-06-07T06:55:24Z
))
```

Accessing S3 data using credentials vended by S3 Access Grants

After a grantee obtains temporary credentials through their access grant, they can use these temporary credentials to call Amazon S3 API operations to access your data.

Grantees can access S3 data by using the AWS Command Line Interface (AWS CLI), the AWS SDKs, and the Amazon S3 REST API. Additionally, you can use the AWS Python and Java plugins to call S3 Access Grants

Using the AWS CLI

After the grantee obtains their temporary credentials from S3 Access Grants, they can set up a profile with these credentials to retrieve the data.

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example commands, replace the *user input placeholders* with your own information.

Example– Set up a profile

```
aws configure set aws_access_key_id "$accessKey" --profile access-grants-consumer-access-profile
aws configure set aws_secret_access_key "$secretKey" --profile access-grants-consumer-access-profile
aws configure set aws_session_token "$sessionToken" --profile access-grants-consumer-access-profile
```

To use the following example command, replace the *user input placeholders* with your own information.

Example– Get the S3 data

The grantee can use the get-object AWS CLI command to access the data. The grantee can also use put-object, ls, and other S3 AWS CLI commands.

```
aws s3api get-object \
```

```
--bucket amzn-s3-demo-bucket1 \
--key myprefix \
--region us-east-2 \
--profile access-grants-consumer-access-profile
```

Using the AWS SDKs

This section provides examples of how grantees can access your S3 data by using the AWS SDKs.

Java

The following Java code example gets an object from an S3 bucket. For instructions on creating and testing a working sample, see Getting Started in the *AWS SDK for Java Developer Guide*.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ResponseHeaderOverrides;
import com.amazonaws.services.s3.model.S3Object;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;

public class GetObject2 {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String key = "*** Object key ***";

        S3Object fullObject = null, objectPortion = null, headerOverrideObject =
 null;
        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
```

```
            // Get an object and print its contents.
            System.out.println("Downloading an object");
            fullObject = s3Client.getObject(new GetObjectRequest(bucketName, key));
            System.out.println("Content-Type: " +
fullObject.getObjectMetadata().getContentType());
            System.out.println("Content: ");
            displayTextInputStream(fullObject.getObjectContent());

            // Get a range of bytes from an object and print the bytes.
            GetObjectRequest rangeObjectRequest = new GetObjectRequest(bucketName,
key)
                    .withRange(0, 9);
            objectPortion = s3Client.getObject(rangeObjectRequest);
            System.out.println("Printing bytes retrieved.");
            displayTextInputStream(objectPortion.getObjectContent());

            // Get an entire object, overriding the specified response headers, and
print
            // the object's content.
            ResponseHeaderOverrides headerOverrides = new ResponseHeaderOverrides()
                    .withCacheControl("No-cache")
                    .withContentDisposition("attachment; filename=example.txt");
            GetObjectRequest getObjectRequestHeaderOverride = new
GetObjectRequest(bucketName, key)
                    .withResponseHeaders(headerOverrides);
            headerOverrideObject =
s3Client.getObject(getObjectRequestHeaderOverride);
            displayTextInputStream(headerOverrideObject.getObjectContent());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        } finally {
            // To ensure that the network connection doesn't remain open, close any
open
            // input streams.
            if (fullObject != null) {
                fullObject.close();
            }
```

```
            if (objectPortion != null) {
                objectPortion.close();
            }
            if (headerOverrideObject != null) {
                headerOverrideObject.close();
            }
        }
    }

    private static void displayTextInputStream(InputStream input) throws IOException
 {
        // Read the text input stream one line at a time and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
        String line = null;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

Supported S3 actions in S3 Access Grants

A grantee can use the temporary credential vended by S3 Access Grants to perform S3 actions on the S3 data they have access to. The following is a list of allowable S3 actions that a grantee can perform. Which actions are allowable depends on the level of permission granted in the access grant, either `READ`, `WRITE`, or `READWRITE`.

> **Note**
>
> In addition to the Amazon S3 permissions listed below, Amazon S3 can call the AWS Key Management Service (AWS KMS) Decrypt (`kms:decrypt`) `READ` permission or the AWS KMS GenerateDataKey (`kms:generateDataKey`) `WRITE` permission. These permissions don't allow direct access to the AWS KMS key.

S3 IAM action	API action & doc	S3 Access Grants Permission	S3 resource			
s3:GetObject	GetObject	READ	Object			
s3:GetObjectVersion	GetObject	READ	Object			
s3:GetObjectAcl	GetObjectAcl	READ	Object			
s3:GetObjectVersionAcl	GetObjectAcl	READ	Object			
s3:ListMultipartUploads	ListParts	READ	Object			
s3:PutObject	PutObject, CreateMultipartUpload, UploadPart, UploadPartCopy, CompleteMultipartUpload	WRITE	Object			
s3:PutObjectAcl	PutObjectAcl	WRITE	Object			
s3:PutObjectVersionAcl	PutObjectAcl	WRITE	Object			
s3:DeleteObject	DeleteObject	WRITE	Object			

S3 IAM action	API action & doc	S3 Access Grants Permission	S3 resource			
s3:DeleteObjectVersion	DeleteObject	WRITE	Object			
s3:AbortMultipartUpload	AbortMultipartUpload	WRITE	Object			
s3:ListBucket	HeadBucket, ListObjectsV2, ListObjects	READ	Bucket			
s3:ListBucketVersions	ListObjectVersions	READ	Bucket			
s3:ListBucketMultipartUploads	ListMultipartUploads	READ	Bucket			

List the caller's access grants

S3 data owners can use S3 Access Grants to create access grants for AWS Identity and Access Management (IAM) identities or for AWS IAM Identity Center corporate directory identities. IAM identies and IAM Identity Center directory identities can in turn use the `ListCallerAccessGrants` API to list all of the Amazon S3 buckets, prefixes, and objects they can access, as defined by their S3 Access Grants. Use this API to discover all of the S3 data an IAM or directory identity can access through S3 Access Grants.

By default, `ListCallerAccessGrants` returns grants from the S3 Access Grants instance in the AWS account ID you provide. To list grants across your own instance and every instance shared with you through AWS Resource Access Manager (AWS RAM), set the `instanceScope` parameter to

ALL. Each returned grant includes the `AccessGrantsInstanceArn` of the instance it belongs to, so you can tell which account owns it.

You can use this feature to build applications that show the data that is accessible to specific end-users. For example, the AWS Storage Browser for S3, an open source UI component that customers use to access S3 buckets, uses this feature to present end-users with the data that they have access to in Amazon S3, based on their S3 Access Grants. Another example is when building an application for browsing, uploading, or downloading data in Amazon S3, you can use this feature to build a tree structure in your application that an end-user could then browse.

> **Note**
>
> For corporate directory identities, when listing the caller's access grants, S3 Access Grants returns the grants of the IAM identity that is used for the identity-aware session. For more information on identity-aware sessions, see Granting permissions to use identity-aware console sessions in the *AWS Identity and Access Management User Guide*.

The grantee whether an IAM identity, or a corporate directory identity can get a list of their access grants by using the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and the AWS SDKs.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

To use the following example command, replace the *`user input placeholders`* with your own information.

Example List a caller's access grants

Request:

```
aws s3control list-caller-access-grants \
--account-id 111122223333 \
--region us-east-2
--max-results 5
```

Response:

```
{
 "NextToken": "6J9S...",
 "CallerAccessGrantsList": [
  {
   "Permission": "READWRITE",
   "GrantScope": "s3://amzn-s3-demo-bucket/prefix1/sub-prefix1/*",
   "ApplicationArn": "NA"
  },
  {
   "Permission": "READWRITE",
   "GrantScope": "s3://amzn-s3-demo-bucket/prefix1/sub-prefix2/*",
   "ApplicationArn": "ALL"
  },
  {
   "Permission": "READWRITE",
   "GrantScope": "s3://amzn-s3-demo-bucket/prefix1/sub-prefix3/*",
   "ApplicationArn": "arn:aws:sso::111122223333:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d"
  }
 ]
}
```

Example List a caller's access grants for a bucket

You can narrow the scope of the results using the `grantscope` parameter.

Request:

```
aws s3control list-caller-access-grants \
--account-id 111122223333 \
--region us-east-2
--grant-scope "s3://amzn-s3-demo-bucket""
--max-results 1000
```

Response:

```
{
 "NextToken": "6J9S...",
 "CallerAccessGrantsList": [
  {
   "Permission": "READ",
   "GrantScope": "s3://amzn-s3-demo-bucket*",
```

```
    "ApplicationArn": "ALL"
   },
   {
    "Permission": "READ",
    "GrantScope": "s3://amzn-s3-demo-bucket/prefix1/*",
    "ApplicationArn": "arn:aws:sso::111122223333:application/ssoins-
ssoins-1234567890abcdef/apl-abcd1234a1b2c3d"
   }
 ]
}
```

Example List a caller's access grants across all shared instances

You can list all the grants you have access to across the instances shared with you by setting the `instanceScope` parameter to `ALL`.

Request:

```
aws s3control list-caller-access-grants \
--account-id 111122223333 \
--instance-scope ALL \
--region us-east-2 \
--max-results 5
```

Response (grants returned from two different accounts' instances):

```
{
 "CallerAccessGrantsList": [
  {
   "Permission": "READWRITE",
   "GrantScope": "s3://amzn-s3-demo-bucket/prefix1/*",
   "ApplicationArn": "NA",
   "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
  },
  {
   "Permission": "READ",
   "GrantScope": "s3://amzn-s3-demo-other-bucket/reports/*",
   "ApplicationArn": "NA",
   "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:444455556666:access-grants/default"
  }
 ]
}
```

Using the REST API

For information about the Amazon S3 REST API support for getting a list of the API caller's access grants, see ListCallerAccessGrants in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

This section provides an example of how grantees request temporary credentials from S3 Access Grants by using the AWS SDKs.

Java

The following code example returns the API caller's access grants to the S3 data of a particular AWS account. To use this code example, replace the *`user input placeholders`* with your own information.

Example List a caller's access grants

Request:

```
Public void ListCallerAccessGrants() {
 ListCallerAccessGrantsRequest listRequest = ListCallerAccessGrantsRequest.builder()
    .withMaxResults(1000)
    .withGrantScope("s3://")
    .accountId("111122223333");
 ListCallerAccessGrantsResponse listResponse =
 s3control.listCallerAccessGrants(listRequest);
 LOGGER.info("ListCallerAccessGrantsResponse: " + listResponse);
 }
```

Response:

```
ListCallerAccessGrantsResponse(
CallerAccessGrantsList=[
 ListCallerAccessGrantsEntry(
  S3Prefix=s3://amzn-s3-demo-bucket/prefix1/,
  Permission=READ,
  ApplicationArn=ALL
 )
])
```

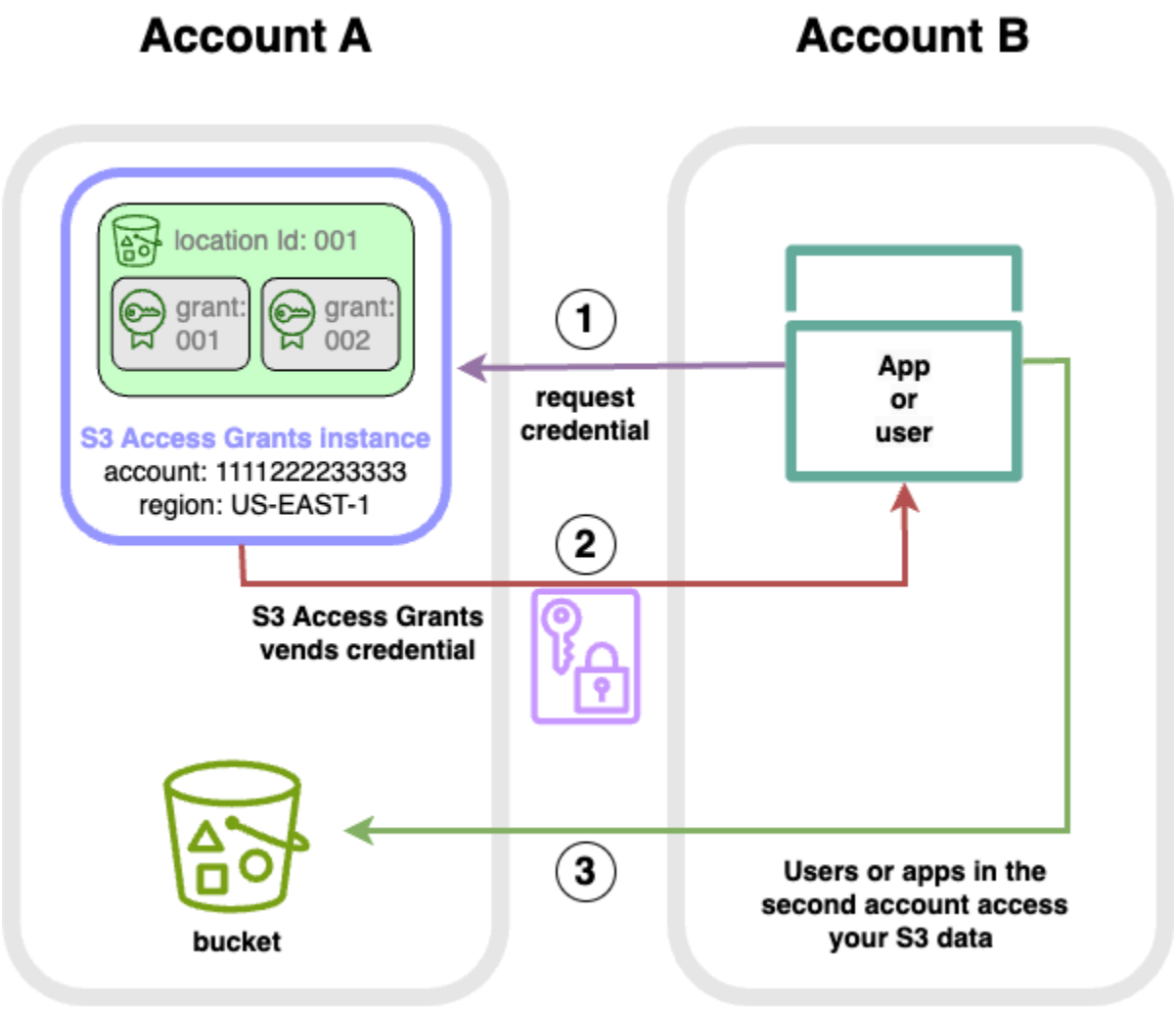
S3 Access Grants cross-account access

With S3 Access Grants, you can grant Amazon S3 data access to the following:

- AWS Identity and Access Management (IAM) identities within your account
- IAM identities in other AWS accounts
- Directory users or groups in your AWS IAM Identity Center instance

First, configure cross-account access for the other account. This includes granting access to your S3 Access Grants instance, either by sharing it through AWS Resource Access Manager (AWS RAM) or by using a resource policy. Then, grant access to your S3 data (buckets, prefixes, or objects) by using grants.

After you configure cross-account access, the other account can request temporary access credentials to your Amazon S3 data from S3 Access Grants. The following image shows the user flow for cross-account S3 access through S3 Access Grants:



1. Users or applications in a second account (B) request credentials from the S3 Access Grants instance in your account (A), where the Amazon S3 data is stored. For more information, see Request access to Amazon S3 data through S3 Access Grants.
2. The S3 Access Grants instance in your account (A) returns temporary credentials if there is a grant that gives the second account access to your Amazon S3 data. For more information on access grants, see Working with grants in S3 Access Grants.
3. Users or applications in the second account (B) use the S3 Access Grants-vended credentials to access the S3 data in your account (A).

Configuring S3 Access Grants cross-account access

To grant cross-account S3 access through S3 Access Grants, follow these steps:

- **Step 1:** Configure an S3 Access Grants instance in your account, for example, account ID 111122223333, where the S3 data is stored.
- **Step 2:** Configure the resource policy for the S3 Access Grants instance in your account 111122223333 to give access to the second account, for example, account ID 444455556666.
- **Step 3:** Configure the IAM permissions for the IAM Principal in the second account 444455556666 to request credentials from the S3 Access Grants instance in your account 111122223333.
- **Step 4:** Create a grant in your account 111122223333 that gives the IAM Principal in the second account 444455556666 access to some of the S3 data in your account 111122223333.

Step 1: Configure an S3 Access Grants instance in your account

First, you must have an S3 Access Grants instance in your account 111122223333 to manage access to your Amazon S3 data. You must create an S3 Access Grants instance in each AWS Region where the S3 data that you want to share is stored. If you are sharing data in more than one AWS Region, then repeat each of these configuration steps for each AWS Region. If you already have an S3 Access Grants instance in the AWS Region where your S3 data is stored, proceed to the next step. If you haven't configured an S3 Access Grants instance, see Working with S3 Access Grants instances to complete this step.

Step 2: Configure the resource policy for your S3 Access Grants instance to grant cross-account access

After you create an S3 Access Grants instance in your account 111122223333 for cross-account access, configure the resource-based policy for the S3 Access Grants instance in your account 111122223333 to grant cross-account access. The S3 Access Grants instance itself supports resource-based policies. With the correct resource-based policy in place, you can grant access for AWS Identity and Access Management (IAM) users or roles from other AWS accounts to your S3 Access Grants instance. Cross-account access only grants these permissions (actions):

- `s3:GetAccessGrantsInstanceForPrefix` — the user, role, or app can retrieve the S3 Access Grants instance that contains a particular prefix.
- `s3:ListAccessGrants`
- `s3:ListAccessLocations`
- `s3:ListCallerAccessGrants`

- `s3:GetDataAccess` — the user, role, or app can request temporary credentials based on the access you were granted through S3 Access Grants. Use these credentials to access the S3 data to which you have been granted access.

You can choose which of these permissions to include in the resource policy. This resource policy on the S3 Access Grants instance is a normal resource-based policy and supports everything that the IAM policy language supports. In the same policy, you can grant access to specific IAM identities in your account `111122223333`, for example, by using the `aws:PrincipalArn` condition, but you don't have to do that with S3 Access Grants. Instead, within your S3 Access Grants instance, you can create grants for individual IAM identities from your account, as well as for the other account. By managing each access grant through S3 Access Grants, you can scale your permissions.

If you already use AWS Resource Access Manager (AWS RAM), you can use it to share your `s3:AccessGrants` resources with other accounts or within your organization. See Working with shared AWS resources for more information. If you don't use AWS RAM, you can also add the resource policy by using the S3 Access Grants API operations or the AWS Command Line Interface (AWS CLI).

Using the S3 console

We recommend that you use the AWS Resource Access Manager (AWS RAM) Console to share your `s3:AccessGrants` resources with other accounts or within your organization. To share S3 Access Grants cross-account, do the following:

To configure the S3 Access Grants instance resource policy:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Select the AWS Region from the AWS Region selector.
3. From the left navigation pane, select **Access Grants**.
4. On the Access Grants instance page, in the **Instance in this account** section, select **Share instance**. This will redirect you to the AWS RAM Console.
5. Select **Create resource share**.
6. Follow the AWS RAM steps to create the resource share. For more information, see Creating a resource share in AWS RAM.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

You can add the resource policy by using the `put-access-grants-instance-resource-policy` CLI command.

If you want to grant cross-account access for the S3 Access Grants instance is in your account `111122223333` to the second account `444455556666`, the resource policy for the S3 Access Grants instance in your account `111122223333` should give the second account `444455556666` permission to perform the following actions:

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

In the S3 Access Grants instance resource policy, specify the ARN of your S3 Access Grants instance as the `Resource`, and the second account `444455556666` as the `Principal`. To use the following example, replace the *`user input placeholders`* with your own information.

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

To add or update the S3 Access Grants instance resource policy, use the following command. When you use the following example command, replace the *user input placeholders* with your own information.

Example Add or update the S3 Access Grants instance resource policy

```
aws s3control put-access-grants-instance-resource-policy \
--account-id 111122223333 \
--policy file://resourcePolicy.json \
--region us-east-2
{
 "Policy": "{\n
   \"Version\": \"2012-10-17\",\n
   \"Statement\": [{\n
  \"Effect\": \"Allow\",\n
  \"Principal\": {\n
    \"AWS\": \"444455556666\"\n
  },\n
  \"Action\": [\n
    \"s3:ListAccessGrants\",\n
    \"s3:ListAccessGrantsLocations\",\n
    \"s3:GetDataAccess\",\n
    \"s3:GetAccessGrantsInstanceForPrefix\",\n
    \"s3:ListCallerAccessGrants"\n
  ],\n
  \"Resource\": \"arn:aws:s3:us-east-2:111122223333:access-grants/default\"\n
    }\n
   ]\n
   }\n",
 "CreatedAt": "2023-06-16T00:07:47.473000+00:00"
}
```

Example Get an S3 Access Grants resource policy

You can also use the CLI to get or delete a resource policy for an S3 Access Grants instance.

To get an S3 Access Grants resource policy, use the following example command. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control get-access-grants-instance-resource-policy \
--account-id 111122223333 \
--region us-east-2
```

```
{
"Policy": "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow
\",\"Principal\":{\"AWS\":\"arn:aws:iam::111122223333:root\"},\"Action\":
[\"s3:ListAccessGrants\",\"s3:ListAccessGrantsLocations\",\"s3:GetDataAccess\",
\"s3:GetAccessGrantsInstanceForPrefix\",\"s3:ListCallerAccessGrants\"],\"Resource\":
\"arn:aws:
s3:us-east-2:111122223333:access-grants/default\"}]}",
"CreatedAt": "2023-06-16T00:07:47.473000+00:00"
}
```

Example Delete an S3 Access Grants resource policy

To delete an S3 Access Grants resource policy, use the following example command. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control delete-access-grants-instance-resource-policy \
--account-id 111122223333 \
--region us-east-2

// No response body
```

Using the REST API

You can add the resource policy by using the PutAccessGrantsInstanceResourcePolicy API.

If you want to grant cross-account access for the S3 Access Grants instance is in your account 111122223333 to the second account 444455556666, the resource policy for the S3 Access Grants instance in your account 111122223333 should give the second account 444455556666 permission to perform the following actions:

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

In the S3 Access Grants instance resource policy, specify the ARN of your S3 Access Grants instance as the `Resource`, and the second account 444455556666 as the `Principal`. To use the following example, replace the *user input placeholders* with your own information.

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

You can then use the PutAccessGrantsInstanceResourcePolicy API to configure the policy.

For information on the REST API support to update, get, or delete a resource policy for an S3 Access Grants instance, see the following sections in the *Amazon Simple Storage Service API Reference*:

- PutAccessGrantsInstanceResourcePolicy
- GetAccessGrantsInstanceResourcePolicy
- DeleteAccessGrantsInstanceResourcePolicy

Using the AWS SDKs

This section provides you with the AWS SDK examples of how to configure your S3 Access Grants resource policy to grant a second AWS account access to some of your S3 data.

Java

Add, update, get, or delete a resource policy to manage cross-account access to your S3 Access Grants instance.

Example Add or update an S3 Access Grants instance resource policy

If you want to grant cross-account access for the S3 Access Grants instance is in your account 111122223333 to the second account 444455556666, the resource policy for the S3 Access Grants instance in your account 111122223333 should give the second account 444455556666 permission to perform the following actions:

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`

In the S3 Access Grants instance resource policy, specify the ARN of your S3 Access Grants instance as the `Resource`, and the second account 444455556666 as the `Principal`. To use the following example, replace the *`user input placeholders`* with your own information.

```
{
"Version": "2012-10-17",
"Statement": [
{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
},
 "Action": [
  "s3:ListAccessGrants",
  "s3:ListAccessGrantsLocations",
  "s3:GetDataAccess",
  "s3:GetAccessGrantsInstanceForPrefix"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
} ]
}
```

To add or update an S3 Access Grants instance resource policy, use the following code example:

```
public void putAccessGrantsInstanceResourcePolicy() {
 PutAccessGrantsInstanceResourcePolicyRequest putRequest =
 PutAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
```

```
 .policy(RESOURCE_POLICY)
 .build();
 PutAccessGrantsInstanceResourcePolicyResponse putResponse =
 s3Control.putAccessGrantsInstanceResourcePolicy(putRequest);
 LOGGER.info("PutAccessGrantsInstanceResourcePolicyResponse: " + putResponse);
 }
```

Response:

```
PutAccessGrantsInstanceResourcePolicyResponse(
 Policy={
 "Version": "2012-10-17",
 "Statement": [{
 "Effect": "Allow",
 "Principal": {
 "AWS": "444455556666"
 },
 "Action": [
 "s3:ListAccessGrants",
 "s3:ListAccessGrantsLocations",
 "s3:GetDataAccess",
 "s3:GetAccessGrantsInstanceForPrefix",
 "s3:ListCallerAccessGrants"
 ],
 "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
 }]
 }
 )
```

Example Get an S3 Access Grants resource policy

To get an S3 Access Grants resource policy, use the following code example. To use the following example command, replace the *`user input placeholders`* with your own information.

```
public void getAccessGrantsInstanceResourcePolicy() {
 GetAccessGrantsInstanceResourcePolicyRequest getRequest =
 GetAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .build();
 GetAccessGrantsInstanceResourcePolicyResponse getResponse =
 s3Control.getAccessGrantsInstanceResourcePolicy(getRequest);
```

```
 LOGGER.info("GetAccessGrantsInstanceResourcePolicyResponse: " + getResponse);
 }
```

Response:

```
GetAccessGrantsInstanceResourcePolicyResponse(
 Policy={"Version": "2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::444455556666:root"},"Action":
["s3:ListAccessGrants","s3:ListAccessGrantsLocations","s3:GetDataAccess","s3:GetAccessGrants
east-2:111122223333:access-grants/default"}]},
 CreatedAt=2023-06-15T22:54:44.319Z
 )
```

Example Delete an S3 Access Grants resource policy

To delete an S3 Access Grants resource policy, use the following code example. To use the following example command, replace the *user input placeholders* with your own information.

```
public void deleteAccessGrantsInstanceResourcePolicy() {
 DeleteAccessGrantsInstanceResourcePolicyRequest deleteRequest =
 DeleteAccessGrantsInstanceResourcePolicyRequest.builder()
 .accountId(111122223333)
 .build();
 DeleteAccessGrantsInstanceResourcePolicyResponse deleteResponse =
 s3Control.putAccessGrantsInstanceResourcePolicy(deleteRequest);
 LOGGER.info("DeleteAccessGrantsInstanceResourcePolicyResponse: " + deleteResponse);
 }
```

Response:

```
DeleteAccessGrantsInstanceResourcePolicyResponse()
```

Step 3: Grant IAM identities in a second account permission to call the S3 Access Grants instance in your account

After the owner of the Amazon S3 data has configured the cross-account policy for the S3 Access Grants instance in account 111122223333, the owner of the second account 444455556666 must create an identity-based policy for its IAM users or roles, and the owner must give them access to

the S3 Access Grants instance. In the identity-based policy, include one or more of the following actions, depending on what's granted in the S3 Access Grants instance resource policy and the permissions you want to grant:

- `s3:ListAccessGrants`
- `s3:ListAccessGrantsLocations`
- `s3:GetDataAccess`
- `s3:GetAccessGrantsInstanceForPrefix`
- `s3:ListCallerAccessGrants`

Following the AWS cross-account access pattern, the IAM users or roles in the second account 444455556666 must explicitly have one or more of these permissions. For example, grant the `s3:GetDataAccess` permission so that the IAM user or role can call the S3 Access Grants instance in account 111122223333 to request credentials.

To use this example command, replace the *`user input placeholders`* with your own information.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
  "Effect": "Allow",
  "Action": [
   "s3:GetDataAccess"
  ],
   "Resource": "arn:aws:s3:us-east-2:111122223333:access-grants/default"
  }
 ]
}
```

For information on editing IAM identity-based policy, see Editing IAM policies in the *AWS Identity and Access Management guide*.

Step 4: Create a grant in the S3 Access Grants instance of your account that gives the IAM identity in the second account access to some of your S3 data

For the final configuration step, you can create a grant in the S3 Access Grants instance in your account 111122223333 that gives access to the IAM identity in the second account 444455556666

to some of the S3 data in your account. You can do this by using the Amazon S3 Console, CLI, API, and SDKs. For more information, see Create grants.

In the grant, specify the AWS ARN of the IAM identity from the second account, and specify which location in your S3 data (a bucket, prefix, or object) that you are granting access to. This location must already be registered with your S3 Access Grants instance. For more information, see Register a location. You can optionally specify a subprefix. For example, if the location you are granting access to is a bucket, and you want to limit the access further to a specific object in that bucket, then pass the object key name in the `S3SubPrefix` field. Or if you want to limit access to the objects in the bucket with key names that start with a specific prefix, such as `2024-03-research-results/`, then pass `S3SubPrefix=2024-03-research-results/`.

The following is an example CLI command for creating an access grant for an identity in the second account. See Create grants for more information. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-access-grant \
--account-id 111122223333 \
--access-grants-location-id default \
--access-grants-location-configuration S3SubPrefix=prefixA* \
--permission READ \
--grantee GranteeType=IAM,GranteeIdentifier=arn:aws:iam::444455556666:role/data-consumer-1
```

After configuring cross-account access, the user or role in the second account can do the following:

- Calls `ListAccessGrantsInstances` to list the S3 Access Grants instances shared with it through AWS RAM. For more information, see Get the details of an S3 Access Grants instance.
- Calls `ListCallerAccessGrants` with `instanceScope` set to `ALL` to list the grants it has access to across its own instance and all instances shared with it through AWS RAM. For more information, see List the caller's access grants.
- Requests temporary credentials from S3 Access Grants. With `instanceScope` set to `ALL` on `GetDataAccess`, it can request credentials for a path in any instance shared with it. The account ID in the request no longer needs to be the account that owns the data. For more information on how to make these requests, see Request access to Amazon S3 data through S3 Access Grants.

Managing tags for S3 Access Grants

Tags in Amazon S3 Access Grants have similar characteristics to object tags in Amazon S3. Each tag is a key-value pair. The resources in S3 Access Grants that you can tag are S3 Access Grants instances, locations, and grants.

> **Note**
>
> Tagging in S3 Access Grants uses different API operations than object tagging. S3 Access Grants uses the TagResource, UntagResource, and ListTagsForResource API operations, where a resource can be either an S3 Access Grants instance, a registered location, or an access grant.

Similar to object tags, the following limitations apply:

- You can add tags to new S3 Access Grants resources when you create them, or you can add tags to existing resources.
- You can associate up to 10 tags with a resource. If multiple tags are associated with the same resource, they must have unique tag keys.
- A tag key can be up to 128 Unicode characters in length, and tag values can be up to 256 Unicode characters in length. Tags are internally represented in UTF-16. In UTF-16, characters consume either 1 or 2 character positions.
- The keys and values are case sensitive.

For more information about tag restrictions, see User-defined tag restrictions in the *AWS Billing User Guide*.

You can tag resources in S3 Access Grants by using the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, or the AWS SDKs.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

You can tag an S3 Access Grants resource when you create it or after you have created it. The following examples show how you tag or untag an S3 Access Grants instance. You can perform similar operations for registered locations and access grants.

To use the following example commands, replace the *user input placeholders* with your own information.

Example– Create an S3 Access Grants instance with tags

```
aws s3control create-access-grants-instance \
 --account-id 111122223333 \
 --profile access-grants-profile \
 --region us-east-2 \
 --tags Key=tagKey1,Value=tagValue1
```

Response:

```
 {
    "CreatedAt": "2023-10-25T01:09:46.719000+00:00",
    "AccessGrantsInstanceId": "default",
    "AccessGrantsInstanceArn": "arn:aws:s3:us-east-2:111122223333:access-grants/
default"
}
```

Example– Tag an already created S3 Access Grants instance

```
aws s3control tag-resource \
--account-id 111122223333 \
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2 \
--tags Key=tagKey2,Value=tagValue2
```

Example– List tags for the S3 Access Grants instance

```
aws s3control list-tags-for-resource \
--account-id 111122223333 \
--resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
--profile access-grants-profile \
--region us-east-2
```

Response:

```
{
```

```
    "Tags": [
        {
            "Key": "tagKey1",
            "Value": "tagValue1"
        },
        {
            "Key": "tagKey2",
            "Value": "tagValue2"
        }
    ]
}
```

Example– Untag the S3 Access Grants instance

```
aws s3control untag-resource \
 --account-id 111122223333 \
 --resource-arn "arn:aws:s3:us-east-2:111122223333:access-grants/default" \
 --profile access-grants-profile \
 --region us-east-2 \
 --tag-keys "tagKey2"
```

Using the REST API

You can use the Amazon S3 API to tag, untag, or list tags for an S3 Access Grants instance, registered location, or access grant. For information about the REST API support for managing S3 Access Grants tags, see the following sections in the *Amazon Simple Storage Service API Reference*:

- TagResource
- UntagResource
- ListTagsForResource

S3 Access Grants limitations

S3 Access Grants has the following limitations:

> **Note**
>
> If your use case exceeds these limitations, contact AWS support to request higher limits.

S3 Access Grants instance

You can create **1 S3 Access Grants instance** per AWS Region per account. See Create an S3 Access Grants instance.

S3 Access Grants location

You can register **1,000 S3 Access Grants locations** per S3 Access Grants instance. See Working with S3 Access Grants locations.

Grant

You can create **100,000 grants** per S3 Access Grants instance. See Create a grant.

S3 Access Grants AWS Regions

S3 Access Grants is currently available in the following AWS Regions:

AWS Region code	AWS Region name
us-east-1	US East (N. Virginia)
us-east-2	US East (Ohio)
us-west-1	US West (N. California)
us-west-2	US West (Oregon)
af-south-1	Africa (Cape Town)
ap-east-1	Asia Pacific (Hong Kong)
ap-east-2	Asia Pacific (Taipei)
ap-northeast-1	Asia Pacific (Tokyo)
ap-northeast-2	Asia Pacific (Seoul)
ap-northeast-3	Asia Pacific (Osaka)
ap-south-1	Asia Pacific (Mumbai)

AWS Region code	AWS Region name
ap-south-2	Asia Pacific (Hyderabad)
ap-southeast-1	Asia Pacific (Singapore)
ap-southeast-2	Asia Pacific (Sydney)
ap-southeast-3	Asia Pacific (Jakarta)
ap-southeast-4	Asia Pacific (Melbourne)
ap-southeast-6	Asia Pacific (New Zealand)
ap-southeast-7	Asia Pacific (Thailand)
ca-central-1	Canada (Central)
ca-west-1	Canada West (Calgary)
eu-central-1	Europe (Frankfurt)
eu-central-2	Europe (Zurich)
eu-north-1	Europe (Stockholm)
eu-south-1	Europe (Milan)
eu-south-2	Europe (Spain)
eu-west-1	Europe (Ireland)
eu-west-2	Europe (London)
eu-west-3	Europe (Paris)
eusc-de-east-1	AWS European Sovereign Cloud (Germany)
il-central-1	Israel (Tel Aviv)
me-central-1	Middle East (UAE)

AWS Region code	AWS Region name
me-south-1	Middle East (Bahrain)
mx-central-1	Mexico (Central)
sa-east-1	South America (São Paulo)
us-gov-east-1	AWS GovCloud (US-East)
us-gov-west-1	AWS GovCloud (US-West)

S3 Access Grants integrations

S3 Access Grants can be used with the following AWS services and features. This page will be updated as new integrations become available.

> **Tip**
>
> This AWS workshop for S3 Access Grants walks you through using S3 Access Grants with AWS Identity and Access Management (IAM) users, IAM Identity Center users, Amazon EMR, and AWS Transfer Family.

Amazon Athena

- Using IAM Identity Center enabled Athena workgroups

Amazon EMR

- Launch an Amazon EMR cluster with S3 Access Grants

Amazon EMR on EKS

- Launch an Amazon EMR on EKS cluster with S3 Access Grants

Amazon EMR Serverless application

- Launch an Amazon EMR Serverless application with S3 Access Grants

Amazon Redshift

- Amazon Redshift integration with Amazon S3 Access Grants

Amazon SageMaker AI Studio

Adding Amazon S3 data to Amazon SageMaker AI Unified Studio

Using S3 Access Grants in Amazon SageMaker AI Unified Studio, you can share your Amazon S3 data in multiple projects. To enable granting access to data using S3 Access Grants, an S3 Access Grants instance is required. Amazon SageMaker AI Unified Studio will use an S3 Access Grants instance if one is already available or can create an instance. First, you add your Amazon S3 data and then publish the data to the catalog or share it directly with consumers.

Using Amazon S3 Access Grants with Amazon SageMaker AI Studio and the SDK for Python (Boto3) plugin

Using S3 Access Grants in Amazon SageMaker AI Studio notebooks is now easier when using the SDK for Python (Boto3) plugin. Set up access grants for IAM principals and AWS IAM Identity Center directory users, beforehand. Although Amazon SageMaker AI Studio does not natively support identity provider directory users, you can write custom Python code, using the plugin that allows these identities to access S3 data through S3 Access Grants. The data access is taking place with the help of the plugin and not through Amazon SageMaker AI.

AWS Glue

Amazon S3 Access Grants with AWS Glue

AWS IAM Identity Center

Trusted identity propagation across applications

AWS Transfer Family

Configure Amazon S3 Access Grants for AWS Transfer Family

Storage Browser for S3

Managing data access at scale using Storage Browser for S3

Open source Python frameworks

Amazon S3 Access Grants now integrates with open source Python frameworks

Managing access with ACLs

Access control lists (ACLs) are one of the resource-based options that you can use to manage access to your buckets and objects. You can use ACLs to grant basic read/write permissions to other AWS accounts. There are limits to managing permissions using ACLs.

For example, you can grant permissions only to other AWS accounts; you cannot grant permissions to users in your account. You cannot grant conditional permissions, nor can you explicitly deny permissions. ACLs are suitable for specific scenarios. For example, if a bucket owner allows other AWS accounts to upload objects, permissions to these objects can only be managed using object ACL by the AWS account that owns the object.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

> **Important**
>
> If your general purpose bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your general purpose bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

For more information about ACLs, see the following topics.

Topics

- Access control list (ACL) overview
- Configuring ACLs

- Policy examples for ACLs

Access control list (ACL) overview

Amazon S3 access control lists (ACLs) enable you to manage access to buckets and objects. Each bucket and object has an ACL attached to it as a subresource. It defines which AWS accounts or groups are granted access and the type of access. When a request is received against a resource, Amazon S3 checks the corresponding ACL to verify that the requester has the necessary access permissions.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

> **Important**
>
> If your general purpose bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your general purpose bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

When you create a bucket or an object, Amazon S3 creates a default ACL that grants the resource owner full control over the resource. This is shown in the following sample bucket ACL (the default object ACL has the same structure):

Example

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>*** Owner-Canonical-User-ID ***</ID>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:type="Canonical User">
        <ID>*** Owner-Canonical-User-ID ***</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

The sample ACL includes an `Owner` element that identifies the owner by the AWS account's canonical user ID. For instructions on finding your canonical user ID, see Finding an AWS account canonical user ID. The `Grant` element identifies the grantee (either an AWS account or a predefined group) and the permission granted. This default ACL has one `Grant` element for the owner. You grant permissions by adding `Grant` elements, with each grant identifying the grantee and the permission.

> **Note**
>
> An ACL can have up to 100 grants.

Topics

- Who is a grantee?
- What permissions can I grant?
- aclRequired values for common Amazon S3 requests
- Sample ACL
- Canned ACL

Who is a grantee?

When you grant access rights, you specify each grantee as a *type*="*value*" pair, where *type* is one of the following:

- `id` – If the value specified is the canonical user ID of an AWS account
- `uri` – If you are granting permissions to a predefined group

> **Warning**
>
> When you grant other AWS accounts access to your resources, be aware that the AWS accounts can delegate their permissions to users under their accounts. This is known as *cross-account access*. For information about using cross-account access, see Creating a Role to Delegate Permissions to an IAM User in the *IAM User Guide*.

Finding an AWS account canonical user ID

The canonical user ID is associated with your AWS account. This ID is a long string of characters, such as:

`79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be`

For information about how to find the canonical user ID for your account, see Find the canonical user ID for your AWS account in the *AWS Account Management Reference Guide*.

You can also look up the canonical user ID of an AWS account by reading the ACL of a bucket or an object to which the AWS account has access permissions. When an individual AWS account is granted permissions by a grant request, a grant entry is added to the ACL with the account's canonical user ID.

> **Note**
>
> If you make your bucket public (not recommended), any unauthenticated user can upload objects to the bucket. These anonymous users don't have an AWS account. When an anonymous user uploads an object to your bucket, Amazon S3 adds a special canonical user ID (`65a011a29cdf8ec533ec3d1ccaae921c`) as the object owner in the ACL. For more information, see Amazon S3 bucket and object ownership.

Amazon S3 predefined groups

Amazon S3 has a set of predefined groups. When granting account access to a group, you specify one of the Amazon S3 URIs instead of a canonical user ID. Amazon S3 provides the following predefined groups:

- **Authenticated Users group** – Represented by `http://acs.amazonaws.com/groups/global/AuthenticatedUsers`.

  This group represents all AWS accounts. **Access permission to this group allows any AWS account to access the resource.** However, all requests must be signed (authenticated).

  > **Warning**
  >
  > When you grant access to the **Authenticated Users group**, any AWS authenticated user in the world can access your resource.

- **All Users group** – Represented by `http://acs.amazonaws.com/groups/global/AllUsers`.

  Access permission to this group allows anyone in the world access to the resource. The requests can be signed (authenticated) or unsigned (anonymous). Unsigned requests omit the Authentication header in the request.

  > **Warning**
  >
  > We highly recommend that you never grant the **All Users group** `WRITE`, `WRITE_ACP`, or `FULL_CONTROL` permissions. For example, although `WRITE` permissions deny non-owners the ability to overwrite or delete existing objects, `WRITE` permissions still allow anyone to store objects in your bucket, for which you are billed. For more details about these permissions, see the following section What permissions can I grant?.

- **Log Delivery group** – Represented by `http://acs.amazonaws.com/groups/s3/LogDelivery`.

  `WRITE` permission on a bucket enables this group to write server access logs (see Logging requests with server access logging) to the bucket.

> **Note**
>
> When using ACLs, a grantee can be an AWS account or one of the predefined Amazon S3 groups. However, the grantee cannot be an IAM user. For more information about AWS users and permissions within IAM, see Using AWS Identity and Access Management.

What permissions can I grant?

The following table lists the set of permissions that Amazon S3 supports in an ACL. The set of ACL permissions is the same for an object ACL and a bucket ACL. However, depending on the context (bucket ACL or object ACL), these ACL permissions grant permissions for specific buckets or object operations. The table lists the permissions and describes what they mean in the context of objects and buckets.

For more information about ACL permissions in the Amazon S3 console, see Configuring ACLs.

Permission	When granted on a bucket	When granted on an object
`READ`	Allows grantee to list the objects in the bucket	Allows grantee to read the object data and its metadata
`WRITE`	Allows grantee to create new objects in the bucket. For the bucket and object owners of existing objects, also allows deletions and overwrites of those objects	Not applicable
`READ_ACP`	Allows grantee to read the bucket ACL	Allows grantee to read the object ACL
`WRITE_ACP`	Allows grantee to write the ACL for the applicable bucket	Allows grantee to write the ACL for the applicable object
`FULL_CONTROL`	Allows grantee the `READ`, `WRITE`, `READ_ACP`, and `WRITE_ACP` permissions on the bucket	Allows grantee the `READ`, `READ_ACP`, and `WRITE_ACP` permissions on the object

Warning

Use caution when granting access permissions to your S3 buckets and objects. For example, granting `WRITE` access to a bucket allows the grantee to create objects in the bucket. We highly recommend that you read through the entire Access control list (ACL) overview section before granting permissions.

Mapping of ACL permissions and access policy permissions

As shown in the preceding table, an ACL allows only a finite set of permissions, compared to the number of permissions that you can set in an access policy (see Policy actions for Amazon S3). Each of these permissions allows one or more Amazon S3 operations.

The following table shows how each ACL permission maps to the corresponding access policy permissions. As you can see, access policy allows more permissions than an ACL does. You use ACLs primarily to grant basic read/write permissions, similar to file system permissions. For more information about when to use an ACL, see Identity and Access Management for Amazon S3.

For more information about ACL permissions in the Amazon S3 console, see Configuring ACLs.

ACL permission	Corresponding access policy permissions when the ACL permission is granted on a bucket	Corresponding access policy permissions when the ACL permission is granted on an object
`READ`	`s3:ListBucket` , `s3:ListBucketVersions` , and `s3:ListBucketMultipartUploads`	`s3:GetObject` and `s3:GetObjectVersion`
`WRITE`	`s3:PutObject` Bucket owner can create, overwrite, and delete any object in the bucket, and object owner has `FULL_CONTROL` over their object. In addition, when the grantee is the bucket owner, granting `WRITE` permission in a bucket ACL allows	Not applicable

ACL permission	Corresponding access policy permissions when the ACL permission is granted on a bucket	Corresponding access policy permissions when the ACL permission is granted on an object
	the `s3:DeleteObjectVersion` action to be performed on any version in that bucket.	
`READ_ACP`	`s3:GetBucketAcl`	`s3:GetObjectAcl` and `s3:GetObjectVersionAcl`
`WRITE_ACP`	`s3:PutBucketAcl`	`s3:PutObjectAcl` and `s3:PutObjectVersionAcl`
`FULL_CONTROL`	Equivalent to granting `READ`, `WRITE`, `READ_ACP`, and `WRITE_ACP` ACL permissions. Accordingly, this ACL permission maps to a combination of corresponding access policy permissions.	Equivalent to granting `READ`, `READ_ACP`, and `WRITE_ACP` ACL permissions. Accordingly, this ACL permission maps to a combination of corresponding access policy permissions.

Condition keys

When you grant access policy permissions, you can use condition keys to constrain the value for the ACL on an object using a bucket policy. The following context keys correspond to ACLs. You can use these context keys to mandate the use of a specific ACL in a request:

- `s3:x-amz-grant-read` - Require read access.
- `s3:x-amz-grant-write` - Require write access.
- `s3:x-amz-grant-read-acp` - Require read access to the bucket ACL.
- `s3:x-amz-grant-write-acp` - Require write access to the bucket ACL.
- `s3:x-amz-grant-full-control` - Require full control.
- `s3:x-amz-acl` - Require a Canned ACL.

For example policies that involve ACL-specific headers, see Granting s3:PutObject permission with a condition requiring the bucket owner to get full control. For a complete list of Amazon S3

specific condition keys, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

`aclRequired` values for common Amazon S3 requests

To identify Amazon S3 requests that required ACLs for authorization, you can use the `aclRequired` value in Amazon S3 server access logs or AWS CloudTrail. The `aclRequired` value that appears in CloudTrail or Amazon S3 server access logs depends on which operations were called and certain information about the requester, object owner, and bucket owner. If no ACLs were required, or if you are setting the `bucket-owner-full-control` canned ACL, or if the requests are allowed by your bucket policy, the `aclRequired` value string is "`-`" in Amazon S3 server access logs and is absent in CloudTrail.

The following tables list the expected `aclRequired` values in CloudTrail or Amazon S3 server access logs for the various Amazon S3 API operations. You can use this information to understand which Amazon S3 operations depend on ACLs for authorization. In the following tables, A, B, and C represent the different accounts associated with the requester, object owner, and bucket owner. Entries with an asterisk (*) indicate any of accounts A, B, or C.

> **Note**
>
> `PutObject` operations in the following table, unless specified otherwise, indicate requests that do not set an ACL, unless the ACL is a `bucket-owner-full-control` ACL. A null value for `aclRequired` indicates that `aclRequired` is absent in AWS CloudTrail logs.

The following table shows the `aclRequired` values for CloudTrail.

Operation name	Requester	Object owner	Bucket owner	Bucket policy grants access	`aclRequired` value	Reason
`GetObject`	A	A	A	Yes or No	null	Same-account access

Operation name	Requester	Object owner	Bucket owner	Bucket policy grants access	aclRequired value	Reason
GetObject	A	B	A	Yes or No	null	Same-account access with bucket owner enforced
GetObject	A	A	B	Yes	null	Cross-account access granted by bucket policy
GetObject	A	A	B	No	Yes	Cross-account access relies on ACL
GetObject	A	A	B	Yes	null	Cross-account access granted by bucket policy
GetObject	A	B	B	No	Yes	Cross-account access relies on ACL

Operation name	Requester	Object owner	Bucket owner	Bucket policy grants access	aclRequired value	Reason
GetObject	A	B	C	Yes	null	Cross-account access granted by bucket policy
GetObject	A	B	C	No	Yes	Cross-account access relies on ACL
PutObject	A	Not applicable	A	Yes or No	null	Same-account access
PutObject	A	Not applicable	B	Yes	null	Cross-account access granted by bucket policy
PutObject	A	Not applicable	B	No	Yes	Cross-account access relies on ACL

Operation name	Requester	Object owner	Bucket owner	Bucket policy grants access	aclRequired value	Reason
PutObject with an ACL (except for bucket-owner-full-control)	*	Not applicable	*	Yes or No	Yes	Request grants ACL
ListObjects	A	Not applicable	A	Yes or No	null	Same-account access
ListObjects	A	Not applicable	B	Yes	null	Cross-account access granted by bucket policy
ListObjects	A	Not applicable	B	No	Yes	Cross-account access relies on ACL
DeleteObject	A	Not applicable	A	Yes or No	null	Same-account access

Operation name	Requester	Object owner	Bucket owner	Bucket policy grants access	aclRequired value	Reason
DeleteObject	A	Not applicable	B	Yes	null	Cross-account access granted by bucket policy
DeleteObject	A	Not applicable	B	No	Yes	Cross-account access relies on ACL
PutObjectAcl	*	*	*	Yes or No	Yes	Request grants ACL
PutBucketAcl	*	Not applicable	*	Yes or No	Yes	Request grants ACL

Note

REST.PUT.OBJECT operations in the following table, unless specified otherwise, indicate requests that do not set an ACL, unless the ACL is a bucket-owner-full-control ACL. An aclRequired value string of "-" indicates a null value in Amazon S3 server access logs.

The following table shows the aclRequired values for Amazon S3 server access logs.

Operation name	Requester	Object owner	Bucket owner	Bucket policy grants access	aclRequired value	Reason
REST.GET.OBJECT	A	A	A	Yes or No	-	Same-account access
REST.GET.OBJECT	A	B	A	Yes or No	-	Same-account access with bucket owner enforced
REST.GET.OBJECT	A	A	B	Yes	-	Cross-account access granted by bucket policy
REST.GET.OBJECT	A	A	B	No	Yes	Cross-account access relies on ACL
REST.GET.OBJECT	A	B	B	Yes	-	Cross-account access granted by bucket policy
REST.GET.OBJECT	A	B	B	No	Yes	Cross-account access

Operation name	Requester	Object owner	Bucket owner	Bucket policy grants access	aclRequired value	Reason
						relies on ACL
REST.GET.OBJECT	A	B	C	Yes	-	Cross-account access granted by bucket policy
REST.GET.OBJECT	A	B	C	No	Yes	Cross-account access relies on ACL
REST.PUT.OBJECT	A	Not applicable	A	Yes or No	-	Same-account access
REST.PUT.OBJECT	A	Not applicable	B	Yes	-	Cross-account access granted by bucket policy
REST.PUT.OBJECT	A	Not applicable	B	No	Yes	Cross-account access relies on ACL

Operation name	Requester	Object owner	Bucket owner	Bucket policy grants access	aclRequired value	Reason
REST.PUT.OBJECT with an ACL (except for bucket-owner-full-control)	*	Not applicable	*	Yes or No	Yes	Request grants ACL
REST.GET.BUCKET	A	Not applicable	A	Yes or No	-	Same-account access
REST.GET.BUCKET	A	Not applicable	B	Yes	-	Cross-account access granted by bucket policy
REST.GET.BUCKET	A	Not applicable	B	No	Yes	Cross-account access relies on ACL
REST.DELETE.OBJECT	A	Not applicable	A	Yes or No	-	Same-account access

Operation name	Requester	Object owner	Bucket owner	Bucket policy grants access	aclRequired value	Reason
REST.DELETE.OBJECT	A	Not applicable	B	Yes	-	Cross-account access granted by bucket policy
REST.DELETE.OBJECT	A	Not applicable	B	No	Yes	Cross-account access relies on ACL
REST.PUT.ACL	*	*	*	Yes or No	Yes	Request grants ACL

Sample ACL

The following sample ACL on a bucket identifies the resource owner and a set of grants. The format is the XML representation of an ACL in the Amazon S3 REST API. The bucket owner has FULL_CONTROL of the resource. In addition, the ACL shows how permissions are granted on a resource to two AWS accounts, identified by canonical user ID, and two of the predefined Amazon S3 groups discussed in the preceding section.

Example

```
<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Owner>
    <ID>Owner-canonical-user-ID</ID>
  </Owner>
  <AccessControlList>
    <Grant>
```

```
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>Owner-canonical-user-ID</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>user1-canonical-user-ID</ID>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="CanonicalUser">
        <ID>user2-canonical-user-ID</ID>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>

    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
      </Grantee>
      <Permission>WRITE</Permission>
    </Grant>

  </AccessControlList>
</AccessControlPolicy>
```

Canned ACL

Amazon S3 supports a set of predefined grants, known as *canned ACLs*. Each canned ACL has a predefined set of grantees and permissions. The following table lists the set of canned ACLs and the associated predefined grants.

Canned ACL	Applies to	Permissions added to ACL
`private`	Bucket and object	Owner gets `FULL_CONTROL`. No one else has access rights (default).
`public-read`	Bucket and object	Owner gets `FULL_CONTROL`. The `AllUsers` group (see Who is a grantee?) gets `READ` access.
`public-read-write`	Bucket and object	Owner gets `FULL_CONTROL`. The `AllUsers` group gets `READ` and `WRITE` access. Granting this on a bucket is generally not recommended.
`aws-exec-read`	Bucket and object	Owner gets `FULL_CONTROL`. Amazon EC2 gets `READ` access to `GET` an Amazon Machine Image (AMI) bundle from Amazon S3.
`authenticated-read`	Bucket and object	Owner gets `FULL_CONTROL`. The `AuthenticatedUsers` group gets `READ` access.
`bucket-owner-read`	Object	Object owner gets `FULL_CONTROL`. Bucket owner gets `READ` access. If you specify this canned ACL when creating a bucket, Amazon S3 ignores it.
`bucket-owner-full-control`	Object	Both the object owner and the bucket owner get `FULL_CONTROL` over the object. If you specify this canned ACL when creating a bucket, Amazon S3 ignores it.

Canned ACL	Applies to	Permissions added to ACL
`log-delivery-write`	Bucket	The `LogDelivery` group gets `WRITE` and `READ_ACP` permissions on the bucket. For more information about logs, see (Logging requests with server access logging).

> **Note**
>
> You can specify only one of these canned ACLs in your request.

You specify a canned ACL in your request by using the `x-amz-acl` request header. When Amazon S3 receives a request with a canned ACL in the request, it adds the predefined grants to the ACL of the resource.

Configuring ACLs

This section explains how to manage access permissions for S3 buckets and objects using access control lists (ACLs). You can add grants to your resource ACL using the AWS Management Console, AWS Command Line Interface (CLI), REST API, or AWS SDKs.

Bucket and object permissions are independent of each other. An object does not inherit the permissions from its bucket. For example, if you create a bucket and grant write access to a user, you can't access that user's objects unless the user explicitly grants you access.

You can grant permissions to other AWS account users or to predefined groups. The user or group that you are granting permissions to is called the *grantee*. By default, the owner, which is the AWS account that created the bucket, has full permissions.

Each permission you grant for a user or group adds an entry in the ACL that is associated with the bucket. The ACL lists grants, which identify the grantee and the permission granted.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled.

When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

Important

If your general purpose bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your general purpose bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

Warning

Do not grant write access to the **Everyone (public access)** or **Authenticated Users group (all AWS authenticated users)** groups. For more information about the effects of granting write access to these groups, see Amazon S3 predefined groups.

Using the S3 console to set ACL permissions for a bucket

The console displays combined access grants for duplicate grantees. To see the full list of ACLs, use the Amazon S3 REST API, AWS CLI, or AWS SDKs.

The following table shows the ACL permissions that you can configure for buckets in the Amazon S3 console.

Amazon S3 console ACL permissions for buckets

Console permission	ACL permission	Access
Objects - List	READ	Allows grantee to list the objects in the bucket.

Console permission	ACL permission	Access
Objects - Write	`WRITE`	Allows grantee to create new objects in the bucket. For the bucket and object owners of existing objects, also allows deletions and overwrites of those objects.
Bucket ACL - Read	`READ_ACP`	Allows grantee to read the bucket ACL.
Bucket ACL - Write	`WRITE_ACP`	Allows grantee to write the ACL for the applicable bucket.
Everyone (public access): Objects - List	`READ`	Grants public read access for the objects in the bucket. When you grant list access to **Everyone (public access)**, anyone in the world can access the objects in the bucket.
Everyone (public access): Bucket ACL - Read	`READ_ACP`	Grants public read access for the bucket ACL. When you grant read access to **Everyone (public access)**, anyone in the world can access the bucket ACL.

For more information about ACL permissions, see Access control list (ACL) overview.

Important

If your general purpose bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your general purpose bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

To set ACL permissions for a bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want to set permissions for.
4. Choose **Permissions**.
5. Under **Access control list**, choose **Edit**.

   You can edit the following ACL permissions for the bucket:

   Objects

   - **List** – Allows a grantee to list the objects in the bucket.
   - **Write** – Allows grantee to create new objects in the bucket. For the bucket and object owners of existing objects, also allows deletions and overwrites of those objects.

     In the S3 console, you can only grant write access to the S3 log delivery group and the bucket owner (your AWS account). Do not grant write access for other grantees. However, if you need to grant write access, you can use the AWS CLI, AWS SDKs, or the REST API.

   Bucket ACL

   - **Read** – Allows grantee to read the bucket ACL.
   - **Write** – Allows grantee to write the ACL for the applicable bucket.
6. To change the bucket owner's permissions, beside **Bucket owner (your AWS account)**, clear or select from the following ACL permissions:

   - **Objects** – **List** or **Write**
   - **Bucket ACL** – **Read** or **Write**

   The *owner* refers to the AWS account root user, not an AWS Identity and Access Management IAM user. For more information about the root user, see The AWS account root user in the *IAM User Guide*.
7. To grant or undo permissions for the general public (everyone on the internet), beside **Everyone (public access)**, clear or select from the following ACL permissions:

- **Objects** – **List**
- **Bucket ACL** – **Read**

> **Warning**
>
> Use caution when granting the **Everyone** group public access to your S3 bucket. When you grant access to this group, anyone in the world can access your bucket. Never grant any kind of public write access to your S3 bucket.

8. To grant or undo permissions for anyone with an AWS account, beside **Authenticated Users group (anyone with an AWS account)**, clear or select from the following ACL permissions:

   - **Objects** – **List**
   - **Bucket ACL** – **Read**

9. To grant or undo permissions for Amazon S3 to write server access logs to the bucket, under **S3 log delivery group**, clear or select from the following ACL permissions:

   - **Objects** – **List** or **Write**
   - **Bucket ACL** – **Read** or **Write**

   If a bucket is set up as the target to receive access logs, the bucket permissions must allow the **Log Delivery** group write access. When you enable server access logging on a bucket, the Amazon S3 console grants write access to the **Log Delivery** group for the target bucket that you choose. For more information about server access logging, see Enabling Amazon S3 server access logging.

10. To grant access to another AWS account, do the following:

    a. Choose **Add grantee**.

    b. In the **Grantee** box, enter the canonical ID of the other AWS account.

    c. Select from the following ACL permissions:

       - **Objects** – **List** or **Write**
       - **Bucket ACL** – **Read** or **Write**

> **Warning**
>
> When you grant other AWS accounts access to your resources, be aware that the AWS accounts can delegate their permissions to users under their accounts. This is known as *cross-account access*. For information about using cross-account access, see Creating a Role to Delegate Permissions to an IAM User in the *IAM User Guide*.

11. To remove access to another AWS account, under **Access for other AWS accounts**, choose **Remove**.
12. To save your changes, choose **Save changes**.

Using the S3 console to set ACL permissions for an object

The console displays combined access grants for duplicate grantees. To see the full list of ACLs, use the Amazon S3 REST API, AWS CLI, or AWS SDKs. The following table shows the ACL permissions that you can configure for objects in the Amazon S3 console.

Amazon S3 console ACL permissions for objects

Console permission	ACL permission	Access
Object - Read	`READ`	Allows grantee to read the object data and its metadata.
Object ACL - Read	`READ_ACP`	Allows grantee to read the object ACL.
Object ACL - Write	`WRITE_ACP`	Allows grantee to write the ACL for the applicable object

For more information about ACL permissions, see Access control list (ACL) overview.

> **Important**
>
> If your general purpose bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your general purpose bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control

lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

To set ACL permissions for an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **objects** list, choose the name of the object for which you want to set permissions.
4. Choose **Permissions**.
5. Under Access control list (ACL), choose **Edit**.

   You can edit the following ACL permissions for the object:

   Object

   - **Read** – Allows grantee to read the object data and its metadata.

   Object ACL

   - **Read** – Allows grantee to read the object ACL.
   - **Write** – Allows grantee to write the ACL for the applicable object. In the S3 console, you can only grant write access to the bucket owner (your AWS account). Do not grant write access for other grantees. However, if you need to grant write access, you can use the AWS CLI, AWS SDKs, or the REST API.
6. You can manage object access permissions for the following:

   a. **Access for object owner**

      The *owner* refers to the AWS account root user, and not an AWS Identity and Access Management IAM user. For more information about the root user, see The AWS account root user in the *IAM User Guide*.

      To change the owner's object access permissions, under **Access for object owner**, choose **Your AWS Account (owner)**.

Select the check boxes for the permissions that you want to change, and then choose **Save**.

b. **Access for other AWS accounts**

To grant permissions to an AWS user from a different AWS account, under **Access for other AWS accounts**, choose **Add account**. In the **Enter an ID** field, enter the canonical ID of the AWS user that you want to grant object permissions to. For information about finding a canonical ID, see Your AWS account identifiers in the *Amazon Web Services General Reference*. You can add as many as 99 users.

Select the check boxes for the permissions that you want to grant to the user, and then choose **Save**. To display information about the permissions, choose the Help icons.

c. **Public access**

To grant access to your object to the general public (everyone in the world), under **Public access**, choose **Everyone**. Granting public access permissions means that anyone in the world can access the object.

Select the check boxes for the permissions that you want to grant, and then choose **Save**.

> **Warning**
>
> - Use caution when granting the **Everyone** group anonymous access to your Amazon S3 objects. When you grant access to this group, anyone in the world can access your object. If you need to grant access to everyone, we highly recommend that you only grant permissions to **Read objects**.
> - *Do not* grant the **Everyone** group write object permissions. Doing so allows anyone to overwrite the ACL permissions for the object.

Using the AWS SDKs

The following examples show how to set access control list (ACL) grants on buckets and objects.

Important

If your general purpose bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your general purpose bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

Java

The following examples show how to set access control list (ACL) grants on buckets and objects. The first example creates a bucket with a canned ACL (see Canned ACL), creates a list of custom grants, and then replaces the canned ACL with the custom grants. The second example shows how to modify an ACL using the `AccessControlList.grantPermission()` method.

Example Create a bucket and specify a canned ACL that grants permission to the S3 log delivery group

This example creates a bucket. In the request, the example specifies a canned ACL that grants the Log Delivery group permission to write logs to the bucket.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import java.io.IOException;
import java.util.ArrayList;

public class CreateBucketWithACL {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String userEmailForReadPermission = "*** user@example.com ***";

        try {
```

```
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .build();

            // Create a bucket with a canned ACL. This ACL will be replaced by the
            // setBucketAcl()
            // calls below. It is included here for demonstration purposes.
            CreateBucketRequest createBucketRequest = new
CreateBucketRequest(bucketName, clientRegion.getName())
                    .withCannedAcl(CannedAccessControlList.LogDeliveryWrite);
            s3Client.createBucket(createBucketRequest);

            // Create a collection of grants to add to the bucket.
            ArrayList<Grant> grantCollection = new ArrayList<Grant>();

            // Grant the account owner full control.
            Grant grant1 = new Grant(new
CanonicalGrantee(s3Client.getS3AccountOwner().getId()),
                    Permission.FullControl);
            grantCollection.add(grant1);

            // Grant the LogDelivery group permission to write to the bucket.
            Grant grant2 = new Grant(GroupGrantee.LogDelivery, Permission.Write);
            grantCollection.add(grant2);

            // Save grants by replacing all current ACL grants with the two we just
created.
            AccessControlList bucketAcl = new AccessControlList();
            bucketAcl.grantAllPermissions(grantCollection.toArray(new Grant[0]));
            s3Client.setBucketAcl(bucketName, bucketAcl);

            // Retrieve the bucket's ACL, add another grant, and then save the new
ACL.
            AccessControlList newBucketAcl = s3Client.getBucketAcl(bucketName);
            Grant grant3 = new Grant(new
EmailAddressGrantee(userEmailForReadPermission), Permission.Read);
            newBucketAcl.grantAllPermissions(grant3);
            s3Client.setBucketAcl(bucketName, newBucketAcl);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
```

```
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example Update ACL on an existing object

This example updates the ACL on an object. The example performs the following tasks:

- Retrieves an object's ACL
- Clears the ACL by removing all existing permissions
- Adds two permissions: full access to the owner, and WRITE_ACP (see What permissions can I grant?) to a user identified by an email address
- Saves the ACL to the object

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.AccessControlList;
import com.amazonaws.services.s3.model.CanonicalGrantee;
import com.amazonaws.services.s3.model.EmailAddressGrantee;
import com.amazonaws.services.s3.model.Permission;

import java.io.IOException;

public class ModifyACLExistingObject {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String emailGrantee = "*** user@example.com ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
```

```
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Get the existing object ACL that we want to modify.
            AccessControlList acl = s3Client.getObjectAcl(bucketName, keyName);

            // Clear the existing list of grants.
            acl.getGrantsAsList().clear();

            // Grant a sample set of permissions, using the existing ACL owner for
 Full
            // Control permissions.
            acl.grantPermission(new CanonicalGrantee(acl.getOwner().getId()),
 Permission.FullControl);
            acl.grantPermission(new EmailAddressGrantee(emailGrantee),
 Permission.WriteAcp);

            // Save the modified ACL back to the object.
            s3Client.setObjectAcl(bucketName, keyName, acl);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

Example Create a bucket and specify a canned ACL that grants permission to the S3 log delivery group

This C# example creates a bucket. In the request, the code also specifies a canned ACL that grants the Log Delivery group permissions to write the logs to the bucket.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ManagingBucketACLTest
    {
        private const string newBucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            CreateBucketUseCannedACLAsync().Wait();
        }

        private static async Task CreateBucketUseCannedACLAsync()
        {
            try
            {
                // Add bucket (specify canned ACL).
                PutBucketRequest putBucketRequest = new PutBucketRequest()
                {
                    BucketName = newBucketName,
                    BucketRegion = S3Region.EUW1, // S3Region.US,
                                                  // Add canned ACL.
                    CannedACL = S3CannedACL.LogDeliveryWrite
                };
                PutBucketResponse putBucketResponse = await
 client.PutBucketAsync(putBucketRequest);

                // Retrieve bucket ACL.
                GetACLResponse getACLResponse = await client.GetACLAsync(new
 GetACLRequest
                {
                    BucketName = newBucketName
                });
```

```
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("S3 error occurred. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

Example Update ACL on an existing object

This C# example updates the ACL on an existing object. The example performs the following tasks:

- Retrieves an object's ACL.
- Clears the ACL by removing all existing permissions.
- Adds two permissions: full access to the owner, and WRITE_ACP to a user identified by email address.
- Saves the ACL by sending a `PutAcl` request.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ManagingObjectACLTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** object key name ***";
```

```
        private const string emailAddress = "*** email address ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            TestObjectACLTestAsync().Wait();
        }
        private static async Task TestObjectACLTestAsync()
        {
            try
            {
                    // Retrieve the ACL for the object.
                    GetACLResponse aclResponse = await client.GetACLAsync(new
GetACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName
                    });

                    S3AccessControlList acl = aclResponse.AccessControlList;

                    // Retrieve the owner (we use this to re-add permissions after
we clear the ACL).
                    Owner owner = acl.Owner;

                    // Clear existing grants.
                    acl.Grants.Clear();

                    // Add a grant to reset the owner's full permission (the
previous clear statement removed all permissions).
                    S3Grant fullControlGrant = new S3Grant
                    {
                        Grantee = new S3Grantee { CanonicalUser = owner.Id },
                        Permission = S3Permission.FULL_CONTROL

                    };

                    // Describe the grant for the permission using an email address.
                    S3Grant grantUsingEmail = new S3Grant
                    {
                        Grantee = new S3Grantee { EmailAddress = emailAddress },
```

```
                        Permission = S3Permission.WRITE_ACP
                    };
                    acl.Grants.AddRange(new List<S3Grant> { fullControlGrant,
 grantUsingEmail });

                    // Set a new ACL.
                    PutACLResponse response = await client.PutACLAsync(new
 PutACLRequest
                    {
                        BucketName = bucketName,
                        Key = keyName,
                        AccessControlList = acl
                    });
            }
            catch (AmazonS3Exception amazonS3Exception)
            {
                Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
 amazonS3Exception.ToString());
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: " + e.ToString());
            }
        }
    }
}
```

Using the REST API

Amazon S3 APIs enable you to set an ACL when you create a bucket or an object. Amazon S3 also provides API to set an ACL on an existing bucket or an object. These APIs provide the following methods to set an ACL:

- **Set ACL using request headers—** When you send a request to create a resource (bucket or object), you set an ACL using the request headers. Using these headers, you can either specify a canned ACL or specify grants explicitly (identifying grantee and permissions explicitly).
- **Set ACL using request body—** When you send a request to set an ACL on an existing resource, you can set the ACL either in the request header or in the body.

For information on the REST API support for managing ACLs, see the following sections in the *Amazon Simple Storage Service API Reference*:

- GetBucketAcl
- PutBucketAcl
- GetObjectAcl
- PutObjectAcl
- PutObject
- CreateBucket
- CopyObject
- CreateMultipartUpload

> **Important**
>
> If your general purpose bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your general purpose bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

Access Control List (ACL)-Specific Request Headers

You can use headers to grant access control list (ACL)-based permissions. By default, all objects are private. Only the owner has full access control. When you add a new object, you can grant permissions to individual AWS accounts or to predefined groups that Amazon S3 defines. These permissions are then added to the ACL on the object. For more information, see Access control list (ACL) overview.

With this operation, you can grant access permissions using one these two methods:

- **Canned ACL (`x-amz-acl`)** — Amazon S3 supports a set of predefined ACLs, known as canned ACLs. Each canned ACL has a predefined set of grantees and permissions. For more information, see Canned ACL.
- **Access Permissions** — To explicitly grant access permissions to specific AWS accounts or groups, use the following headers. Each header maps to specific permissions that Amazon S3 supports in

an ACL. For more information, see Access control list (ACL) overview. In the header, you specify a list of grantees who get the specific permission.

- x-amz-grant-read
- x-amz-grant-write
- x-amz-grant-read-acp
- x-amz-grant-write-acp
- x-amz-grant-full-control

Using the AWS CLI

For more information about managing ACLs using the AWS CLI, see put-bucket-acl in the *AWS CLI Command Reference*.

> **Important**
>
> If your general purpose bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your general purpose bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

Policy examples for ACLs

You can use condition keys in bucket policies to control access to Amazon S3.

Topics

- Granting s3:PutObject permission with a condition requiring the bucket owner to get full control
- Granting s3:PutObject permission with a condition on the x-amz-acl header

Granting s3:PutObject permission with a condition requiring the bucket owner to get full control

The PUT Object operation allows access control list (ACL)–specific headers that you can use to grant ACL-based permissions. Using these keys, the bucket owner can set a condition to require specific access permissions when the user uploads an object.

Suppose that Account A owns a bucket, and the account administrator wants to grant Dave, a user in Account B, permissions to upload objects. By default, objects that Dave uploads are owned by Account B, and Account A has no permissions on these objects. Because the bucket owner is paying the bills, it wants full permissions on the objects that Dave uploads. The Account A administrator can do this by granting the `s3:PutObject` permission to Dave, with a condition that the request include ACL-specific headers that either grant full permission explicitly or use a canned ACL. For more information, see PUT Object.

Require the x-amz-full-control header

You can require the `x-amz-full-control` header in the request with full control permission to the bucket owner. The following bucket policy grants the `s3:PutObject` permission to user Dave with a condition using the `s3:x-amz-grant-full-control` condition key, which requires the request to include the `x-amz-full-control` header.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/Dave"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        }
    ]
}
```

> **Note**
>
> This example is about cross-account permission. However, if Dave (who is getting the permission) belongs to the AWS account that owns the bucket, this conditional permission is not necessary. This is because the parent account to which Dave belongs owns objects that the user uploads.

Add explicit deny

The preceding bucket policy grants conditional permission to user Dave in Account B. While this policy is in effect, it is possible for Dave to get the same permission without any condition through some other policy. For example, Dave can belong to a group, and you grant the group `s3:PutObject` permission without any condition. To avoid such permission loopholes, you can write a stricter access policy by adding explicit deny. In this example, you explicitly deny the user Dave upload permission if he does not include the necessary headers in the request granting full permissions to the bucket owner. Explicit deny always supersedes any other permission granted. The following is the revised access policy example with explicit deny added.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "statement1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        },
        {
            "Sid": "statement2",
```

```
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:user/AccountBadmin"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::awsexamplebucket1/*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-grant-full-control": "id=AccountA-CanonicalUserID"
                }
            }
        }
    ]
}
```

Test the policy with the AWS CLI

If you have two AWS accounts, you can test the policy using the AWS Command Line Interface (AWS CLI). You attach the policy and use Dave's credentials to test the permission using the following AWS CLI `put-object` command. You provide Dave's credentials by adding the `--profile` parameter. You grant full control permission to the bucket owner by adding the `--grant-full-control` parameter. For more information about setting up and using the AWS CLI, see Developing with Amazon S3 using the AWS CLI in the *Amazon S3 API Reference*.

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg
 --grant-full-control id="AccountA-CanonicalUserID" --profile AccountBUserProfile
```

Require the x-amz-acl header

You can require the `x-amz-acl` header with a canned ACL granting full control permission to the bucket owner. To require the `x-amz-acl` header in the request, you can replace the key-value pair in the `Condition` block and specify the `s3:x-amz-acl` condition key, as shown in the following example.

```
"Condition": {
    "StringEquals": {
        "s3:x-amz-acl": "bucket-owner-full-control"
    }
}
```

To test the permission using the AWS CLI, you specify the `--acl` parameter. The AWS CLI then adds the `x-amz-acl` header when it sends the request.

```
aws s3api put-object --bucket examplebucket --key HappyFace.jpg --body c:\HappyFace.jpg
 --acl "bucket-owner-full-control" --profile AccountBadmin
```

Granting s3:PutObject permission with a condition on the x-amz-acl header

The following bucket policy grants the `s3:PutObject` permission for two AWS accounts if the request includes the `x-amz-acl` header making the object publicly readable. The `Condition` block uses the `StringEquals` condition, and it is provided a key-value pair, `"s3:x-amz-acl": ["public-read"]`, for evaluation. In the key-value pair, the `s3:x-amz-acl` is an Amazon S3–specific key, as indicated by the prefix `s3:`.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AddCannedAcl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root",
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::awsexamplebucket1/*"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": [
                        "public-read"
                    ]
                }
            }
        }
    ]
```

```
}
```

Important

Not all conditions make sense for all actions. For example, it makes sense to include an `s3:LocationConstraint` condition on a policy that grants the `s3:CreateBucket` Amazon S3 permission. However, it does not make sense to include this condition on a policy that grants the `s3:GetObject` permission. Amazon S3 can test for semantic errors of this type that involve Amazon S3–specific conditions. However, if you are creating a policy for an IAM user or role and you include a semantically invalid Amazon S3 condition, no error is reported because IAM cannot validate Amazon S3 conditions.

Blocking public access to your Amazon S3 storage

The Amazon S3 Block Public Access feature provides settings for access points, buckets, accounts, and AWS Organizations to help you manage public access to Amazon S3 resources. By default, new buckets, access points, and objects don't allow public access. However, users can modify bucket policies, access point policies, or object permissions to allow public access. S3 Block Public Access settings override these policies and permissions so that you can limit public access to these resources.

With S3 Block Public Access, organization administrators, account administrators, and bucket owners can easily set up centralized controls to limit public access to their Amazon S3 resources that are enforced regardless of how the resources are created.

You can manage Block Public Access settings at multiple levels: organization level (using AWS Organizations), account level, and bucket and access point level. For instructions on configuring public block access, see Configuring block public access.

When Amazon S3 receives a request to access a bucket or an object, it determines whether the bucket or the bucket owner's account has a block public access setting applied. If the account is part of an AWS Organizations with Block Public Access policies, Amazon S3 also checks for organization-level settings. If the request was made through an access point, Amazon S3 also checks for block public access settings for the access point. If there is an existing block public access setting that prohibits the requested access, Amazon S3 rejects the request.

Amazon S3 Block Public Access provides four settings. These settings are independent and can be used in any combination. Each setting can be applied to an access point, a bucket, or an entire AWS account. At the organization level, all four settings are applied together as a unified policy - you cannot select individual settings granularly. If the block public access settings for the access point, bucket, or account differ, then Amazon S3 applies the most restrictive combination of the access point, bucket, and account settings. Account-level settings automatically inherit organization-level policies when present, and S3 takes the most restrictive policy between bucket-level and effective account-level settings. For example, if your organization has a Block Public Access policy enabled, but a specific bucket has Block Public Access disabled at the bucket level, the bucket will still be protected because S3 applies the more restrictive organization/account-level settings. Conversely, if your organization policy is disabled but a bucket has Block Public Access enabled, that bucket remains protected by its bucket-level settings.

When Amazon S3 evaluates whether an operation is prohibited by a block public access setting, it rejects any request that violates an organization policy (which enforces the account BPA setting) or an access point, bucket, or account setting.

Important

Public access is granted to buckets and objects through access control lists (ACLs), access point policies, bucket policies, or all. To help ensure that all of your Amazon S3 access points, buckets, and objects have their public access blocked, we recommend that you turn on all four settings for block public access for your account. For organizations managing multiple accounts, consider using organization-level Block Public Access policies for centralized control. Additionally, we recommend that you also turn on all four settings for each bucket to comply with AWS Security Hub Foundational Security Best Practices control S3.8. These settings block public access for all current and future buckets and access points. Before applying these settings, verify that your applications will work correctly without public access. If you require some level of public access to your buckets or objects, for example to host a static website as described at Hosting a static website using Amazon S3, you can customize the individual settings to suit your storage use cases.
Enabling Block Public Access helps protect your resources by preventing public access from being granted through the resource policies or access control lists (ACLs) that are directly attached to S3 resources. In addition to enabling Block Public Access, carefully inspect the following policies to confirm that they don't grant public access:

- Identity-based policies attached to associated AWS principals (for example, IAM roles)

- Resource-based policies attached to associated AWS resources (for example, AWS Key Management Service (KMS) keys)

Note

- You can enable block public access settings only for organizations, access points, buckets, and AWS accounts. Amazon S3 doesn't support block public access settings on a per-object basis.
- When you apply block public access settings to an account, the settings apply to all AWS Regions globally. The settings might not take effect in all Regions immediately or simultaneously, but they eventually propagate to all Regions.
- When you apply organization-level block public access policies, they automatically propagate to selected member accounts and override account-level settings.

Topics

- Block public access settings
- Managing block public access at organization level
- Performing block public access operations on an access point
- The meaning of "public"
- Using IAM Access Analyzer for S3 to review public buckets
- Permissions
- Configuring block public access
- Configuring block public access settings for your account
- Configuring block public access settings for your S3 buckets

Block public access settings

S3 Block Public Access provides four settings. You can apply these settings in any combination to individual access points, buckets, or entire AWS accounts. At the organization level, you can only enable or disable all four settings together using an "all" or "none" approach - granular control over individual settings is not available. If you apply a setting to an account, it applies to all buckets and

access points that are owned by that account. Account-level settings automatically inherit from organization policies when present. Similarly, if you apply a setting to a bucket, it applies to all access points associated with that bucket.

The policy inheritance and enforcement works as follows:

- Organization-level policies automatically apply to member accounts, enforcing any existing account-level settings
- Account-level setting inherit from organization policies when present, or use locally configured settings when no organization policy exists
- Bucket-level settings operate independently but are subject to enforcement restrictions. S3 applies the most restrictive combination across all applicable levels - organization/account-level and bucket-level settings. This means a bucket inherits the baseline protection from its account (which may be organization-managed), but S3 will enforce whichever configuration is more restrictive between the bucket's settings and the account's effective settings.

The following table contains the available settings.

Name	Description
`BlockPublicAcls`	Setting this option to `TRUE` causes the following behavior: • `PutBucketAcl` and `PutObjectAcl` calls fail if the specified access control list (ACL) is public. • `PutObject` calls fail if the request includes a public ACL. • If this setting is applied to an account, then `PUT Bucket` calls fail if the request includes a public ACL. When this setting is set to `TRUE`, the specified operations fail (whether made through the REST API, AWS CLI, or AWS SDKs). However, existing policies and ACLs for buckets and objects aren't modified. This setting enables you to protect against public access while allowing you to audit, refine, or otherwise alter the existing policies and ACLs for your buckets and objects.

Name	Description
	Note Access points don't have ACLs associated with them. If you apply this setting to an access point, it acts as a passthrough to the underlying bucket. If an access point has this setting enabled, requests made through the access point behave as though the underlying bucket has this setting enabled, regardless of whether the bucket actually has this setting enabled.
`IgnorePublicAcls`	Setting this option to TRUE causes Amazon S3 to ignore all public ACLs on a bucket and any objects that it contains. This setting enables you to safely block public access granted by ACLs while still allowing `PutObject` calls that include a public ACL (as opposed to `BlockPublicAcls`, which rejects `PutObject` calls that include a public ACL). Enabling this setting doesn't affect the persistence of any existing ACLs and doesn't prevent new public ACLs from being set. **Note** Access points don't have ACLs associated with them. If you apply this setting to an access point, it acts as a passthrough to the underlying bucket. If an access point has this setting enabled, requests made through the access point behave as though the underlying bucket has this setting enabled, regardless of whether the bucket actually has this setting enabled.

Name	Description
`BlockPublicPolicy`	Setting this option to TRUE for a bucket causes Amazon S3 to reject calls to `PutBucketPolicy` if the specified bucket policy allows public access. Setting this option to TRUE for a bucket also causes Amazon S3 to reject calls to `PutAccessPointPolicy` for all of the bucket's same-account access points if the specified policy allows public access. Setting this option to TRUE for an access point causes Amazon S3 to reject calls to `PutAccessPointPolicy` and `PutBucketPolicy` that are made through the access point if the specified policy (for either the access point or the underlying bucket) allows public access. You can use this setting to allow users to manage access point and bucket policies without allowing them to publicly share the bucket or the objects it contains. Enabling this setting doesn't affect existing access point or bucket policies. **Important** To use this setting effectively, we recommend that you apply it at the *account* level. A bucket policy can allow users to alter a bucket's block public access settings. Therefore, users who have permission to change a bucket policy could insert a policy that allows them to disable the block public access settings for the bucket. If this setting is enabled for the entire account, rather than for a specific bucket, Amazon S3 blocks public policies even if a user alters the bucket policy to disable this setting.

Name	Description
RestrictPublicBuckets	Setting this option to TRUE restricts access to an access point or bucket with a public policy to only AWS service principals and authorized users within the bucket owner's account and access point owner's account. This setting blocks all cross-account access to the access point or bucket (except by AWS service principals), while still allowing users within the account to manage the access point or bucket. Enabling this setting doesn't affect existing access point or bucket policies, except that Amazon S3 blocks public and cross-account access derived from any public access point or bucket policy, including non-public delegation to specific accounts.

Important

- Calls to `GetBucketAcl` and `GetObjectAcl` always return the effective permissions in place for the specified bucket or object. For example, suppose that a bucket has an ACL that grants public access, but the bucket also has the `IgnorePublicAcls` setting enabled. In this case, `GetBucketAcl` returns an ACL that reflects the access permissions that Amazon S3 is enforcing, rather than the actual ACL that is associated with the bucket.
- Block public access settings don't alter existing policies or ACLs. Therefore, removing a block public access setting causes a bucket or object with a public policy or ACL to again be publicly accessible.

Managing block public access at organization level

Organization-level block public access uses AWS Organizations policies to centrally manage S3 public access controls across your entire organization. When enabled, these policies automatically apply to selected accounts and override individual account-level settings.

For additional information on block public access at an organization level, see S3 policy in the *AWS Organizations user guide*.

Performing block public access operations on an access point

To perform block public access operations on an access point, use the AWS CLI service `s3control`.

> **Important**
>
> You can't change an access point's block public access settings after creating the access point. You can specify block public access settings for an access point only when creating the access point.

The meaning of "public"

ACLs

Amazon S3 considers a bucket or object ACL public if it grants any permissions to members of the predefined `AllUsers` or `AuthenticatedUsers` groups. For more information about predefined groups, see Amazon S3 predefined groups.

Bucket policies

When evaluating a bucket policy, Amazon S3 begins by assuming that the policy is public. It then evaluates the policy to determine whether it qualifies as non-public. To be considered non-public, a bucket policy must grant access only to fixed values (values that don't contain a wildcard or an AWS Identity and Access Management Policy Variable) for one or more of the following:

- An AWS principal, user, role, or service principal (e.g. `aws:PrincipalOrgID`)
- A set of Classless Inter-Domain Routings (CIDR) blocks, using `aws:SourceIp`. For more information about CIDR, see RFC 4632 on the RFC Editor website.

> **Note**
>
> Bucket policies that grant access conditioned on the `aws:SourceIp` condition key with very broad IP ranges (for example, 0.0.0.0/1) are evaluated as "public." This includes values broader than /8 for IPv4 and /32 for IPv6 (excluding RFC1918 private ranges). Block public access will reject these "public" policies and prevent cross-account access to buckets that are already using these "public" policies.

- `aws:SourceArn`
- `aws:SourceVpc`
- `aws:SourceVpce`
- `aws:SourceOwner`
- `aws:SourceAccount`
- `aws:userid`, outside the pattern "`AROLEID:*`"
- `s3:DataAccessPointArn`

> **Note**
>
> When used in a bucket policy, this value can contain a wildcard for the access point name without rendering the policy public, as long as the account ID is fixed. For example, allowing access to `arn:aws:s3:us-west-2:123456789012:accesspoint/*` would permit access to any access point associated with account `123456789012` in Region `us-west-2`, without rendering the bucket policy public. This behavior is different for access point policies. For more information, see Access points.

- `s3:DataAccessPointAccount`

For more information about bucket policies, see Bucket policies for Amazon S3.

> **Note**
>
> When using multivalued context keys, you must use the `ForAllValues` or `ForAnyValue` set operators.

Example: Public bucket policies

Under these rules, the following example policies are considered public.

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow"
 }
```

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": { "StringLike": {"aws:SourceVpc": "vpc-*"}}
 }
```

You can make these policies non-public by including any of the condition keys listed previously, using a fixed value. For example, you can make the last policy preceding non-public by setting `aws:SourceVpc` to a fixed value, like the following.

```
{
  "Principal": "*",
  "Resource": "*",
  "Action": "s3:PutObject",
  "Effect": "Allow",
  "Condition": {"StringEquals": {"aws:SourceVpc": "vpc-91237329"}}
 }
```

How Amazon S3 evaluates a bucket policy that contains both public and non-public access grants

This example shows how Amazon S3 evaluates a bucket policy that contains both public and non-public access grants.

Suppose that a bucket has a policy that grants access to a set of fixed principals.
Under the previously described rules, this policy isn't public. Thus, if you enable the `RestrictPublicBuckets` setting, the policy remains in effect as written, because `RestrictPublicBuckets` only applies to buckets that have public policies. However, if you add a public statement to the policy, `RestrictPublicBuckets` takes effect on the bucket. It allows only AWS service principals and authorized users of the bucket owner's account to access the bucket.

As an example, suppose that a bucket owned by "Account-1" has a policy that contains the following:

1. A statement that grants access to AWS CloudTrail (which is an AWS service principal)
2. A statement that grants access to account "Account-2"

3. A statement that grants access to the public, for example by specifying `"Principal": "*"` with no limiting `Condition`

This policy qualifies as public because of the third statement. With this policy in place and `RestrictPublicBuckets` enabled, Amazon S3 allows access only by CloudTrail. Even though statement 2 isn't public, Amazon S3 disables access by "Account-2." This is because statement 3 renders the entire policy public, so `RestrictPublicBuckets` applies. As a result, Amazon S3 disables cross-account access, even though the policy delegates access to a specific account, "Account-2." But if you remove statement 3 from the policy, then the policy doesn't qualify as public, and `RestrictPublicBuckets` no longer applies. Thus, "Account-2" regains access to the bucket, even if you leave `RestrictPublicBuckets` enabled.

Access points

Amazon S3 evaluates block public access settings slightly differently for access points compared to buckets. The rules that Amazon S3 applies to determine when an access point policy is public are generally the same for access points as for buckets, except in the following situations:

- An access point that has a VPC network origin is always considered non-public, regardless of the contents of its access point policy.
- An access point policy that grants access to a set of access points using `s3:DataAccessPointArn` is considered public. Note that this behavior is different than for bucket policies. For example, a bucket policy that grants access to values of `s3:DataAccessPointArn` that match `arn:aws:s3:us-west-2:123456789012:accesspoint/*` is not considered public. However, the same statement in an access point policy would render the access point public.

Using IAM Access Analyzer for S3 to review public buckets

You can use IAM Access Analyzer for S3 to review buckets with bucket ACLs, bucket policies, or access point policies that grant public access. IAM Access Analyzer for S3 alerts you to buckets that are configured to allow access to anyone on the internet or other AWS accounts, including AWS accounts outside of your organization. For each public or shared bucket, you receive findings that report the source and level of public or shared access.

In IAM Access Analyzer for S3, you can block all public access to a bucket with a single click. You can also drill down into bucket-level permission settings to configure granular levels of access. For

specific and verified use cases that require public or shared access, you can acknowledge and record your intent for the bucket to remain public or shared by archiving the findings for the bucket.

In rare cases, IAM Access Analyzer for S3 and Amazon S3 block public access evaluation might differ on whether a bucket is public. This behavior occurs because Amazon S3 block public access performs validation on the existence of actions in addition to evaluating public access. Suppose that the bucket policy contains an `Action` statement that allows public access for an action that isn't supported by Amazon S3 (for example, `s3:NotASupportedAction`). In this case, Amazon S3 block public access evaluates the bucket as public because such a statement could potentially make the bucket public if the action later becomes supported. In cases where Amazon S3 block public access and IAM Access Analyzer for S3 differ in their evaluations, we recommend reviewing the bucket policy and removing any unsupported actions.

For more information about IAM Access Analyzer for S3, see Reviewing bucket access using IAM Access Analyzer for S3.

Permissions

To use Amazon S3 Block Public Access features, you must have the following permissions.

Operation	Required permissions
GET bucket policy status	`s3:GetBucketPolicyStatus`
GET bucket Block Public Access settings	`s3:GetBucketPublicAccessBlock`
PUT bucket Block Public Access settings	`s3:PutBucketPublicAccessBlock`
DELETE bucket Block Public Access settings	`s3:PutBucketPublicAccessBlock`
GET account Block Public Access settings	`s3:GetAccountPublicAccessBlock`
PUT account Block Public Access settings	`s3:PutAccountPublicAccessBlock`
DELETE account Block Public Access settings	`s3:PutAccountPublicAccessBlock`
PUT access point Block Public Access settings	`s3:CreateAccessPoint`

Note

The DELETE operations require the same permissions as the PUT operations. There are no separate permissions for the DELETE operations.

Configuring block public access

For more information about configuring block public access for your AWS account, your Amazon S3 buckets, and your access points, see the following topics:

- Configuring block public access settings for your account
- Configuring block public access settings for your S3 buckets
- Performing block public access operations on an access point

Configuring block public access settings for your account

Important

If your account is managed by an organization-level Block Public Access policy, you cannot modify these account-level settings. Organization-level policies override account-level configurations. For more information on centralized management options, see S3 policy in the *AWS Organizations user guide*.

Amazon S3 Block Public Access provides settings for access points, buckets, organizations, and accounts to help you manage public access to Amazon S3 resources. By default, new buckets, access points, and objects do not allow public access. For more information, see Blocking public access to your Amazon S3 storage.

Note

Account level settings override settings on individual objects. Configuring your account to block public access will override any public access settings made to individual objects within your account. When organization-level policies are active, account-level settings automatically inherit from the organization policy and cannot be modified directly.

You can use the S3 console, AWS CLI, AWS SDKs, and REST API to configure block public access settings for all the buckets in your account when not managed by organization policies. For more information, see the sections below.

To configure block public access settings for your buckets, see Configuring block public access settings for your S3 buckets. For information about access points, see Performing block public access operations on an access point.

Using the S3 console

Amazon S3 block public access prevents the application of any settings that allow public access to data within S3 buckets. This section describes how to edit block public access settings for all the S3 buckets in your AWS account. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

To edit block public access settings for all the S3 buckets in an AWS account

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Account and organization settings**.
3. Under **Block Public Access settings for this account**:
4. Choose **Edit** to change the block public access settings for all the buckets in your AWS account.
5. Choose the settings that you want to change, and then choose **Save changes**.
6. When you're asked for confirmation, enter **`confirm`**. Then choose **Confirm** to save your changes.

If you receive an error message that says, "This account does not allow changes to its account-level S3 Block Public Access settings due to an organizational S3 Block Public Access policy in effect," your account is managed by organization-level policies. Contact your organization administrator to modify these settings.

Using the AWS CLI

You can use Amazon S3 Block Public Access through the AWS CLI. For more information about setting up and using the AWS CLI, see What is the AWS Command Line Interface?

Account

To perform block public access operations on an account, use the AWS CLI service `s3control`. The account-level operations that use this service are as follows:

- `PutPublicAccessBlock` (for an account)
- `GetPublicAccessBlock` (for an account)
- `DeletePublicAccessBlock` (for an account)

> **Note**
>
> `PutPublicAccessBlock` and `DeletePublicAccessBlock` operations will return an "Access Denied" error when the account is managed by organization-level policies. Account-level `GetPublicAccessBlock` operations will return the enforced organization-level policy if present.

For additional information and examples, see put-public-access-block in the *AWS CLI Reference*.

Using the AWS SDKs

Java

The following examples show you how to use Amazon S3 Block Public Access with the AWS SDK for Java to put a public access block configuration on an Amazon S3 account.

> **Note**
>
> `PutPublicAccessBlock` and `DeletePublicAccessBlock` operations will fail with an "Access Denied" error if the account is managed by organization-level policies.

```
AWSS3ControlClientBuilder controlClientBuilder =
 AWSS3ControlClientBuilder.standard();
controlClientBuilder.setRegion(<region>);
controlClientBuilder.setCredentials(<credentials>);

AWSS3Control client = controlClientBuilder.build();
client.putPublicAccessBlock(new PutPublicAccessBlockRequest()
  .withAccountId(<account-id>)
```

```
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
    .withIgnorePublicAcls(<value>)
    .withBlockPublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

Important

This example pertains only to account-level operations, which use the `AWSS3Control` client class. For bucket-level operations, see the preceding example.

Other SDKs

For information about using the other AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Using the REST API

For information about using Amazon S3 Block Public Access through the REST APIs, see the following topics in the *Amazon Simple Storage Service API Reference*.

- Account-level operations
  - PutPublicAccessBlock - Fails when account is managed by organization policies
  - GetPublicAccessBlock - Returns effective configuration including organization policies.
  - DeletePublicAccessBlock - Fails when account is managed by organization policies.

You'll see following error message for restricted operations: "This account does not allow changes to its account-level S3 Block Public Access settings due to an organizational S3 Block Public Access policy in effect."

Configuring block public access settings for your S3 buckets

Amazon S3 Block Public Access provides settings for access points, buckets, organizations, and accounts to help you manage public access to Amazon S3 resources. By default, new buckets, access points, and objects do not allow public access. For more information, see Blocking public access to your Amazon S3 storage.

> **Note**
>
> Bucket-level Block Public Access settings work alongside organization and account-level policies. S3 applies the most restrictive setting between bucket-level and effective account-level configurations (which may be enforced by organization policies if present).

You can use the S3 console, AWS CLI, AWS SDKs, and REST API to grant public access to one or more buckets. You can also block public access to buckets that are already public. For more information, see the sections below.

To configure block public access settings for every bucket in your account, see Configuring block public access settings for your account. For organization-wide centralized management, see S3 policy in the *AWS Organizations user guide*.

For information about configuring block public access for access points, see Performing block public access operations on an access point.

Using the S3 console

Amazon S3 Block Public Access prevents the application of any settings that allow public access to data within S3 buckets. This section describes how to edit Block Public Access settings for one or more S3 buckets. For information about blocking public access using the AWS CLI, AWS SDKs, and the Amazon S3 REST APIs, see Blocking public access to your Amazon S3 storage.

You can see if your bucket is publicly accessible from the **Buckets** list, in the **IAM Access Analyzer** column. For more information, see Reviewing bucket access using IAM Access Analyzer for S3.

If you see an `Error` when you list your buckets and their public access settings, you might not have the required permissions. Check to make sure you have the following permissions added to your user or role policy:

```
s3:GetAccountPublicAccessBlock
s3:GetBucketPublicAccessBlock
s3:GetBucketPolicyStatus
s3:GetBucketLocation
s3:GetBucketAcl
s3:ListAccessPoints
s3:ListAllMyBuckets
```

In some rare cases, requests can also fail because of an AWS Region outage.

To edit the Amazon S3 block public access settings for a single S3 bucket

Follow these steps if you need to change the public access settings for a single S3 bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Bucket name** list, choose the name of the bucket that you want.
3. Choose **Permissions**.
4. Choose **Edit** next to **Block public access (bucket settings)** to change the public access settings for the bucket. For more information about the four Amazon S3 Block Public Access Settings, see Block public access settings.
5. Choose one of the settings, and then choose **Save changes**.
6. When you're asked for confirmation, enter **`confirm`**. Then choose **Confirm** to save your changes.

> **Important**
>
> Even if you disable bucket-level Block Public Access settings, your bucket may still be protected by account-level or organization-level policies. S3 always applies the most restrictive combination of settings across all levels.

You can also change Amazon S3 Block Public Access settings when you create a bucket. For more information, see Creating a general purpose bucket.

Using the AWS CLI

To block public access on a bucket or to delete the public access block, use the AWS CLI service `s3api`. The bucket-level operations that use this service are as follows:

- `PutPublicAccessBlock` (for a bucket)
- `GetPublicAccessBlock` (for a bucket)
- `DeletePublicAccessBlock` (for a bucket)
- `GetBucketPolicyStatus`

For more information and examples, see put-public-access-block in the *AWS CLI Reference*.

Note

These bucket-level operations are not restricted by organization-level policies. However, the effective public access behavior will still be governed by the most restrictive combination of bucket, account, and organization settings. For more information about the hierarchy and policy interactions, see the section called "Editing bucket public access settings".

Using the AWS SDKs

Java

```
AmazonS3 client = AmazonS3ClientBuilder.standard()
   .withCredentials(<credentials>)
   .build();

client.setPublicAccessBlock(new SetPublicAccessBlockRequest()
  .withBucketName(<bucket-name>)
  .withPublicAccessBlockConfiguration(new PublicAccessBlockConfiguration()
    .withBlockPublicAcls(<value>)
    .withIgnorePublicAcls(<value>)
    .withBlockPublicPolicy(<value>)
    .withRestrictPublicBuckets(<value>)));
```

Important

This example pertains only to bucket-level operations, which use the `AmazonS3` client class. For account-level operations, see the following example.

Other SDKs

For information about using the other AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Using the REST API

For information about using Amazon S3 Block Public Access through the REST APIs, see the following topics in the *Amazon Simple Storage Service API Reference*.

- Bucket-level operations
  - PutPublicAccessBlock
  - GetPublicAccessBlock
  - DeletePublicAccessBlock
  - GetBucketPolicyStatus

Granting public access to your Amazon S3 data

Amazon S3 provides multiple ways to grant public access to your data. You can grant public access through a bucket policy, through object tags combined with a bucket policy, or through object access control lists (ACLs). The mechanism you use depends on your goal. For example, to make an entire bucket publicly accessible, you can use a bucket policy. To make only specific objects accessible, you can add a tag to those objects and make them public with a bucket policy.

By default, Amazon S3 blocks public access. New buckets have Block Public Access turned on and have Object Ownership set to Bucket owner enforced, which disables ACLs. Before any object in a bucket can be publicly accessible, you must turn off the Block Public Access settings for the bucket and the account. For more information, see Blocking public access to your Amazon S3 storage.

If you want to grant temporary access to a single object, you can use a presigned URL. A presigned URL gives limited time access to a single object, and you can control how long the URL remains valid by setting its expiration time. For more information, see Sharing objects with presigned URLs.

> **Important**
>
> When you turn off Block Public Access and grant public access, anyone on the internet can access your objects. Grant public access only when a specific use case requires it.

Topics

- Granting public access to an entire bucket
- Granting public access to specific objects with object tags

- Granting public access to an individual object with an ACL

Granting public access to an entire bucket

You can grant public access to an entire bucket with a bucket policy. This is useful when all the objects that you want to make public can be stored together, such as the files for a static website. Store those objects in a bucket that holds only public content, and keep private data in a separate bucket. We recommend this approach because it keeps a clear boundary between public and private data.

To make the objects in a bucket publicly readable, turn off Block Public Access for the bucket and add a bucket policy that grants everyone the `s3:GetObject` permission.

Example

The following bucket policy grants public read access to every object in the bucket.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
    ]
}
```

Granting public access to specific objects with object tags

You can grant public access to specific objects in a bucket by applying an object tag to them and adding a bucket policy that grants read access to any object with that tag. We recommend this approach when you must keep both private and public objects in the same bucket.

With object tags, a single condition in the bucket policy matches every object that has the tag, so one statement can make a large number of objects public. This approach avoids listing individual object keys in the policy, which can contain characters that a policy does not accept and can exceed the bucket policy size limit.

To grant public access to specific objects, turn off Block Public Access for the bucket, apply an object tag such as `public=true` to each object that you want to make public, and add a bucket policy that grants `s3:GetObject` to objects with that tag. For more information about tagging objects, see Tagging your objects.

> **Note**
>
> Object tags incur additional charges, both to apply them and to store them. For more information, see Amazon S3 pricing.

Example

The following bucket policy grants public read access to every object that has the `public=true` tag.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadTaggedObjects",
            "Effect": "Allow",
            "Principal": "*",
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:ExistingObjectTag/public": "true"
                }
            }
        }
    ]
}
```

Granting public access to an individual object with an ACL

Object ACLs are a legacy way to grant access, and new buckets disable them by default. We recommend that you keep ACLs disabled, and use Amazon S3 resource-based policies (bucket policies and access point policies) or IAM policies instead, unless you must control access for each object individually. Policies provide simpler and more flexible access control. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

> **Important**
>
> Enabling ACLs applies to the entire bucket, not only the object that you make public. After ACLs are enabled, another account that uploads an object can own it, so you might not control every object in your bucket.

To make an object public with an ACL, turn off Block Public Access for the bucket and enable ACLs by changing the Object Ownership setting from Bucket owner enforced to Bucket owner preferred. Then apply the `public-read` ACL to the object, which grants read access to everyone, so anyone can read the object. For more information, see Managing access with ACLs.

Reviewing bucket access using IAM Access Analyzer for S3

IAM Access Analyzer for S3 provides external access findings for your S3 general purpose buckets that are configured to allow access to anyone on the internet (public) or other AWS accounts, including AWS accounts outside of your organization. For each bucket that's shared publicly or with other AWS accounts, you receive findings into the source and level of shared access. For example, IAM Access Analyzer for S3 might show that a bucket has read or write access provided through a bucket access control list (ACL), a bucket policy, a Multi-Region Access Point policy, or an access point policy. With these findings, you can take immediate and precise corrective action to restore your bucket access to what you intended.

The Amazon S3 console presents an **External access summary** in the S3 console next to your list of general purpose buckets. In the summary, you can choose the active findings for each AWS Region to see the details of the finding in the IAM Access Analyzer for S3 page. External acces findings in **External access summary** are automatically updated once every 24 hours.

When reviewing a bucket that allows public access, on the IAM Access Analyzer for S3 page, you can block all public access to the bucket with a single click. We recommend that you block all public access to your buckets unless you require public access to support a specific use case. Before you block all public access, ensure that your applications will continue to work correctly without public access. For more information, see Blocking public access to your Amazon S3 storage.

You can also drill down into bucket-level permission settings to configure granular levels of access. For specific and verified use cases that require public access, such as static website hosting, public downloads, or cross-account sharing, you can acknowledge and record your intent for the bucket to remain public or shared by archiving the findings for the bucket. You can revisit and modify

these bucket configurations at any time. You can also download your findings as a CSV report for auditing purposes.

IAM Access Analyzer for S3 is available at no extra cost on the Amazon S3 console. IAM Access Analyzer for S3 is powered by AWS Identity and Access Management (IAM) IAM Access Analyzer. To use IAM Access Analyzer for S3 in the Amazon S3 console, you must visit the IAM console and create an external access analyzer on a per-Region basis.

For more information about IAM Access Analyzer, see What is IAM Access Analyzer? in the *IAM User Guide*. For more information about IAM Access Analyzer for S3, review the following sections.

Important

- IAM Access Analyzer for S3 requires an account-level analyzer in each AWS Region where you have buckets. To use IAM Access Analyzer for S3, you must visit IAM Access Analyzer and create an analyzer that has an account as the zone of trust. For more information, see Enabling IAM Access Analyzer in *IAM User Guide*.
- IAM Access Analyzer for S3 doesn't analyze the access point policy that's attached to cross-account access points. This behavior occurs because the access point and its policy are outside the zone of trust, that is, the account. Buckets that delegate access to a cross-account access point are listed under **Buckets with public access** if you haven't applied the `RestrictPublicBuckets` block public access setting to the bucket or account. When you apply the `RestrictPublicBuckets` block public access setting, the bucket is reported under **Buckets with access from other AWS accounts — including third-party AWS accounts**.
- When a bucket policy or bucket ACL is added or modified, IAM Access Analyzer generates and updates findings based on the change within 30 minutes. Findings related to account level block public access settings might not be generated or updated for up to 6 hours after you change the settings. Findings related to Multi-Region Access Points might not be generated or updated for up to six hours after the Multi-Region Access Point is created, deleted, or you change its policy.

Topics

- Reviewing a global summary of policies that grant external access to buckets
- Information provided by IAM Access Analyzer for S3

- Blocking all public access
- Reviewing and changing bucket access
- Archiving bucket findings
- Activating an archived bucket finding
- Viewing finding details
- Downloading an IAM Access Analyzer for S3 report

Reviewing a global summary of policies that grant external access to buckets

You can use the **External access summary** to view a global summary of policies that grant external access to buckets across your AWS account directly from the S3 console. This summary helps you identify Amazon S3 general purpose buckets in any AWS Region that allow public access or access from other AWS accounts without needing to inspect policies in each AWS Region individually.

Enabling external access summary and IAM Access Analyzer for S3

To use the **External access summary** and IAM Access Analyzer for S3, you must complete the following prerequisite steps.

1. Grant the required permissions. For more information, see Permissions Required to use IAM Access Analyzer in the *IAM User Guide*.
2. Visit IAM to create an account-level analyzer for each Region where you want to use IAM Access Analyzer.

   You can do this by creating an analyzer that has an account as the zone of trust in the IAM console or by using the AWS CLI or AWS SDKs. For more information, see Enabling IAM Access Analyzer in *IAM User Guide*.

Viewing buckets that allow external access

The **External access summary** displays findings and errors for external access that are provided by IAM Access Analyzer for S3 for general purpose buckets. Archived findings and findings for unused access are not included in the summary but can be viewed in the IAM console or IAM Access Analyzer for S3. For more information, see View the IAM Access Analyzer findings dashboard in the IAM User Guide.

> **Note**
>
> The **External access summary** only includes findings for external access analyzers for each of your AWS accounts, not your AWS Organization.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation panel, choose **General purpose buckets**.
3. Expand the **External access summary**. The console displays active public and cross-account access findings.

   > **Note**
   >
   > If S3 experiences an issue loading bucket details, refresh the general purpose buckets list or view findings in IAM Access Analyzer for S3. For more information, see Reviewing bucket access using IAM Access Analyzer for S3.

4. To view a list of findings or errors for an AWS Region, choose the link to the Region. The IAM Access Analyzer for S3 page displays names of buckets that can be accessed publicly or by other AWS accounts. For more information, see Information provided by IAM Access Analyzer for S3.

Updating access controls for buckets that allow external access

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation panel, choose **General purpose buckets**.
3. Expand the **External access summary**. The console displays active findings for buckets that can be accessed publicly or by other AWS accounts.

   > **Note**
   >
   > If S3 experiences an issue loading bucket details, refresh the general purpose buckets list or view findings in IAM Access Analyzer for S3. For more information, see Reviewing bucket access using IAM Access Analyzer for S3.

4. To view a list of findings or errors for an AWS Region, choose the link to the Region. The IAM Access Analyzer for S3 displays active findings for buckets that can be accessed publicly or by other AWS accounts.

   > **Note**
   >
   > External acces findings in **External access summary** are automatically updated once every 24 hours.

5. To block all public access for a bucket, see Blocking all public access. To change the bucket access, see Reviewing and changing bucket access.

Information provided by IAM Access Analyzer for S3

IAM Access Analyzer for S3 provides findings for buckets that can be accessed outside your AWS account. Buckets that are listed under **Public access findings** can be accessed by anyone on the internet. If IAM Access Analyzer for S3 identifies public buckets, you also see a warning banner that shows you the number of public buckets in your Region. Buckets listed under **Cross-account access findings** are shared conditionally with other AWS accounts, including accounts outside of your organization.

For each bucket, IAM Access Analyzer for S3 provides the following information:

- **Bucket name**
- **Shared through** - How the bucket is shared—through a bucket policy, a bucket ACL, a Multi-Region Access Point policy, or an access point policy. Multi-Region Access Points and cross-account access points are reflected under access points. A bucket can be shared through both policies and ACLs. If you want to find and review the source for your bucket access, you can use the information in this column as a starting point for taking immediate and precise corrective action.
- **Status** - The status of the bucket finding. IAM Access Analyzer for S3 displays findings for all public and shared buckets.
  - **Active** - Finding has not been reviewed.
  - **Archived** - Finding has been reviewed and confirmed as intended.
  - **All** - All findings for buckets that are public or shared with other AWS accounts, including AWS accounts outside of your organization.

- **Access level** - Access permissions granted for the bucket:
  - **List** - List resources.
  - **Read** - Read but not edit resource contents and attributes.
  - **Write** - Create, delete, or modify resources.
  - **Permissions** - Grant or modify resource permissions.
  - **Tagging** - Update tags associated with the resource.
- **External principal** - The AWS account outside of your organization with access to the bucket.
- **Resources control policy (RCP) restriction** - The resource control policy (RCP) that applies to the bucket, if applicable. For more information, see Resource control policies (RCPs).

Blocking all public access

If you want to block all access to a bucket in a single click, you can use the **Block all public access** button in IAM Access Analyzer for S3. When you block all public access to a bucket, no public access is granted. We recommend that you block all public access to your buckets unless you require public access to support a specific and verified use case. Before you block all public access, ensure that your applications will continue to work correctly without public access.

If you don't want to block all public access to your bucket, you can edit your block public access settings on the Amazon S3 console to configure granular levels of access to your buckets. For more information, see Blocking public access to your Amazon S3 storage.

In rare cases, IAM Access Analyzer for S3 and Amazon S3 block public access evaluation might differ on whether a bucket is public. This behavior occurs because Amazon S3 block public access performs validation on the existence of actions in addition to evaluating public access. Suppose that the bucket policy contains an `Action` statement that allows public access for an action that isn't supported by Amazon S3 (for example, `s3:NotASupportedAction`). In this case, Amazon S3 block public access evaluates the bucket as public because such a statement could potentially make the bucket public if the action later becomes supported. In cases where Amazon S3 block public access and IAM Access Analyzer for S3 differ in their evaluations, we recommend reviewing the bucket policy and removing any unsupported actions.

To block all public access to a bucket using IAM Access Analyzer for S3

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the navigation pane on the left, under **Dashboards**, choose **Access analyzer for S3**.
3. In IAM Access Analyzer for S3, choose a bucket.
4. Choose **Block all public access**.
5. To confirm your intent to block all public access to the bucket, in **Block all public access (bucket settings)**, enter **`confirm`**.

   Amazon S3 blocks all public access to your bucket. The status of the bucket finding updates to **resolved**, and the bucket disappears from the IAM Access Analyzer for S3 listing. If you want to review resolved buckets, open IAM Access Analyzer on the IAM Console.

Reviewing and changing bucket access

If you did not intend to grant access to the public or other AWS accounts, including accounts outside of your organization, you can modify the bucket ACL, bucket policy, the Multi-Region Access Point policy, or the access point policy to remove the access to the bucket. The **Shared through** column shows all sources of bucket access: bucket policy, bucket ACL, and/or access point policy. Multi-Region Access Points and cross-account access points are reflected under access points.

To review and change a bucket policy, a bucket ACL, a Multi-Region Access Point, or an access point policy

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Access analyzer for S3**.
3. To see whether public access or shared access is granted through a bucket policy, a bucket ACL, a Multi-Region Access Point policy, or an access point policy, look in the **Shared through** column.
4. Under **Buckets**, choose the name of the bucket with the bucket policy, bucket ACL, Multi-Region Access Point policy, or access point policy that you want to change or review.
5. If you want to change or view a bucket ACL:

   a. Choose **Permissions**.

   b. Choose **Access Control List**.

   c. Review your bucket ACL, and make changes as required.

      For more information, see Configuring ACLs.

6. If you want to change or review a bucket policy:

   a. Choose **Permissions**.

   b. Choose **Bucket Policy**.

   c. Review or change your bucket policy as required.

      For more information, see Adding a bucket policy by using the Amazon S3 console.

7. If you want to change or view a Multi-Region Access Point policy:

   a. Choose **Multi-Region Access Point**.

   b. Choose the Multi-Region Access Point name.

   c. Review or change your Multi-Region Access Point policy as required.

      For more information, see Permissions.

8. If you want to review or change an access point policy:

   a. Choose **Access Points for general purpose buckets** or **Access Points for directory buckets**.

   b. Choose the access point name.

   c. Review or change access as required.

      For more information, see Managing your Amazon S3 access points for general purpose buckets.

   If you edit or remove a bucket ACL, a bucket policy, or an access point policy to remove public or shared access, the status for the bucket findings updates to resolved. The resolved bucket findings disappear from the IAM Access Analyzer for S3 listing, but you can view them in IAM Access Analyzer.

Archiving bucket findings

If a bucket grants access to the public or other AWS accounts, including accounts outside of your organization, to support a specific use case (for example, a static website, public downloads, or cross-account sharing), you can archive the finding for the bucket. When you archive bucket findings, you acknowledge and record your intent for the bucket to remain public or shared.

Archived bucket findings remain in your IAM Access Analyzer for S3 listing so that you always know which buckets are public or shared.

To archive bucket findings in IAM Access Analyzer for S3

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **IAM Access Analyzer for S3**.
3. In IAM Access Analyzer for S3, choose an active bucket.
4. To acknowledge your intent for this bucket to be accessed by the public or other AWS accounts, including accounts outside of your organization, choose **Archive**.
5. Enter **`confirm`**, and choose **Archive**.

Activating an archived bucket finding

After you archive findings, you can always revisit them and change their status back to active, indicating that the bucket requires another review.

To activate an archived bucket finding in IAM Access Analyzer for S3

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Access analyzer for S3**.
3. Choose the archived bucket findings.
4. Choose **Mark as active**.

Viewing finding details

If you need to see more information about a finding, you can open the bucket finding details in IAM Access Analyzer on the IAM Console.

To view finding details in IAM Access Analyzer for S3

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Access analyzer for S3**.
3. In IAM Access Analyzer for S3, choose a bucket.
4. Choose **View details**.

The finding details open in IAM Access Analyzer on the IAM Console.

Downloading an IAM Access Analyzer for S3 report

You can download your bucket findings as a CSV report that you can use for auditing purposes. The report includes the same information that you see in IAM Access Analyzer for S3 on the Amazon S3 console.

To download a report

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane on the left, choose **Access analyzer for S3**.
3. In the Region filter, choose the Region.

   IAM Access Analyzer for S3 updates to shows buckets for the chosen Region.
4. Choose **Download report**.

   A CSV report is generated and saved to your computer.

Verifying bucket ownership with bucket owner condition

Amazon S3 bucket owner condition ensures that the buckets you use in your S3 operations belong to the AWS accounts that you expect.

Most S3 operations read from or write to specific S3 buckets. These operations include uploading, copying, and downloading objects, retrieving or modifying bucket configurations, and retrieving or modifying object configurations. When you perform these operations, you specify the bucket that you want to use by including its name with the request. For example, to retrieve an object from S3, you make a request that specifies the name of a bucket and the object key to retrieve from that bucket.

Because Amazon S3 identifies buckets based on their names, an application that uses an incorrect bucket name in a request could inadvertently perform operations against a different bucket than expected. To help avoid unintentional bucket interactions in situations like this, you can use *bucket owner condition*. Bucket owner condition enables you to verify that the target bucket is owned by the expected AWS account, providing an additional layer of assurance that your S3 operations are having the effects you intend.

Topics

- When to use bucket owner condition
- Verifying a bucket owner
- Examples
- Restrictions and limitations

When to use bucket owner condition

We recommend using bucket owner condition whenever you perform a supported S3 operation and know the account ID of the expected bucket owner. Bucket owner condition is available for all S3 object operations and most S3 bucket operations. For a list of S3 operations that don't support bucket owner condition, see Restrictions and limitations.

To see the benefit of using bucket owner condition, consider the following scenario involving AWS customer Bea:

1. Bea develops an application that uses Amazon S3. During development, Bea uses her testing-only AWS account to create a bucket named `bea-data-test`, and configures her application to make requests to `bea-data-test`.
2. Bea deploys her application, but forgets to reconfigure the application to use a bucket in her production AWS account.
3. In production, Bea's application makes requests to `bea-data-test`, which succeed. This results in production data being written to the bucket in Bea's test account.

Bea can help protect against situations like this by using bucket owner condition. With bucket owner condition, Bea can include the AWS account ID of the expected bucket owner in her requests. Amazon S3 then checks the account ID of the bucket owner before processing each request. If the actual bucket owner doesn't match the expected bucket owner, the request fails.

If Bea uses bucket owner condition, the scenario described earlier won't result in Bea's application inadvertently writing to a test bucket. Instead, the requests that her application makes at step 3 will fail with an `Access Denied` error message. By using bucket owner condition, Bea helps eliminate the risk of accidentally interacting with buckets in the wrong AWS account.

Verifying a bucket owner

To use bucket owner condition, you include a parameter with your request that specifies the expected bucket owner. Most S3 operations involve only a single bucket, and require only this single parameter to use bucket owner condition. For `CopyObject` operations, this first parameter specifies the expected owner of the destination bucket, and you include a second parameter to specify the expected owner of the source bucket.

When you make a request that includes a bucket owner condition parameter, S3 checks the account ID of the bucket owner against the specified parameter before processing the request. If the parameter matches the bucket owner's account ID, S3 processes the request. If the parameter doesn't match the bucket owner's account ID, the request fails with an `Access Denied` error message.

You can use bucket owner condition with the AWS Command Line Interface (AWS CLI), AWS SDKs, and Amazon S3 REST APIs. When using bucket owner condition with the AWS CLI and Amazon S3 REST APIs, use the following parameter names.

Access method	Parameter for non-copy operations	Copy operation source parameter	Copy operation destination parameter
AWS CLI	`--expected-bucket-owner`	`--expected-source-bucket-owner`	`--expected-bucket-owner`
Amazon S3 REST APIs	`x-amz-expected-bucket-owner` header	`x-amz-source-expected-bucket-owner` header	`x-amz-expected-bucket-owner` header

The parameter names that are required to use bucket owner condition with the AWS SDKs vary depending on the language. To determine the required parameters, see the SDK documentation for your desired language. You can find the SDK documentation at Tools to Build on AWS.

Examples

The following examples show how you can implement bucket owner condition in Amazon S3 using the AWS CLI or the AWS SDK for Java 2.x.

Example

Example: Upload an object

The following example uploads an object to S3 bucket *amzn-s3-demo-bucket1*, using bucket owner condition to ensure that *amzn-s3-demo-bucket1* is owned by AWS account 111122223333.

AWS CLI

```
aws s3api put-object \
                --bucket amzn-s3-demo-bucket1 --key exampleobject --
body example_file.txt \
                --expected-bucket-owner 111122223333
```

AWS SDK for Java 2.x

```
public void putObjectExample() {
    S3Client s3Client = S3Client.create();;
    PutObjectRequest request = PutObjectRequest.builder()
            .bucket("amzn-s3-demo-bucket1")
            .key("exampleobject")
            .expectedBucketOwner("111122223333")
            .build();
    Path path = Paths.get("example_file.txt");
    s3Client.putObject(request, path);
}
```

Example

Example: Copy an object

The following example copies the object `object1` from S3 bucket *amzn-s3-demo-bucket1* to S3 bucket *amzn-s3-demo-bucket2*. It uses bucket owner condition to ensure that the buckets are owned by the expected accounts according to the following table.

Bucket	Expected owner
amzn-s3-demo-bucket1	111122223333
amzn-s3-demo-bucket2	444455556666

AWS CLI

```
aws s3api copy-object --copy-source amzn-s3-demo-bucket1/object1 \
                            --bucket amzn-s3-demo-bucket2 --key object1copy \
                            --expected-source-bucket-owner 111122223333 --expected-
bucket-owner 444455556666
```

AWS SDK for Java 2.x

```
public void copyObjectExample() {
        S3Client s3Client = S3Client.create();
        CopyObjectRequest request = CopyObjectRequest.builder()
                .copySource("amzn-s3-demo-bucket1/object1")
                .destinationBucket("amzn-s3-demo-bucket2")
                .destinationKey("object1copy")
                .expectedSourceBucketOwner("111122223333")
                .expectedBucketOwner("444455556666")
                .build();
        s3Client.copyObject(request);
    }
```

Example

Example: Retrieve a bucket policy

The following example retrieves the access policy for S3 bucket *amzn-s3-demo-bucket1*, using bucket owner condition to ensure that *amzn-s3-demo-bucket1* is owned by AWS account 111122223333.

AWS CLI

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket1 --expected-bucket-
owner 111122223333
```

AWS SDK for Java 2.x

```
public void getBucketPolicyExample() {
    S3Client s3Client = S3Client.create();
    GetBucketPolicyRequest request = GetBucketPolicyRequest.builder()
            .bucket("amzn-s3-demo-bucket1")
```

```
            .expectedBucketOwner("111122223333")
            .build();
    try {
        GetBucketPolicyResponse response = s3Client.getBucketPolicy(request);
    }
    catch (S3Exception e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    }
}
```

Restrictions and limitations

Amazon S3 bucket owner condition has the following restrictions and limitations:

- The value of the bucket owner condition parameter must be an AWS account ID (12-digit numeric value). Service principals aren't supported.
- Bucket owner condition isn't available for CreateBucket, ListBuckets, or any of the operations included in AWS S3 Control. Amazon S3 ignores any bucket owner condition parameters included with requests to these operations.
- Bucket owner condition only verifies that the account specified in the verification parameter owns the bucket. Bucket owner condition doesn't check the configuration of the bucket. It also doesn't guarantee that the bucket's configuration meets any specific conditions or matches any past state.

Controlling ownership of objects and disabling ACLs for your bucket

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to control ownership of objects uploaded to your bucket and to disable or enable access control lists (ACLs). By default, Object Ownership is set to the Bucket owner enforced setting and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to data exclusively using access management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs, and we recommend that you keep ACLs disabled except in circumstances where you must control access for

each object individually. With ACLs disabled, you can use policies to more easily control access to every object in your bucket, regardless of who uploaded the objects in your bucket.

Object Ownership has three settings that you can use to control ownership of objects uploaded to your bucket and to disable or enable ACLs:

ACLs disabled

- **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect permissions to data in the S3 bucket. The bucket uses policies to define access control.

ACLs enabled

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

For the majority of modern use cases in S3, we recommend that you keep ACLs disabled by applying the Bucket owner enforced setting and using your bucket policy to share data with users outside of your account as needed. This approach simplifies permissions management. You can disable ACLs on both newly created and already existing buckets. For newly created buckets, ACLs are disabled by default. In the case of an existing bucket that already has objects in it, after you disable ACLs, the object and bucket ACLs are no longer part of an access evaluation, and access is granted or denied on the basis of policies. For existing buckets, you can re-enable ACLs at any time after you disable them, and your preexisting bucket and object ACLs are restored.

Before you disable ACLs, we recommend that you review your bucket policy to ensure that it covers all the ways that you intend to grant access to your bucket outside of your account. After you disable ACLs, your bucket accepts only PUT requests that do not specify an ACL or PUT requests with bucket owner full control ACLs, for example, the `bucket-owner-full-control` canned ACL or equivalent forms of this ACL expressed in XML. Existing applications that support bucket owner full control ACLs see no impact. PUT requests that contain other ACLs (for example, custom grants to certain AWS accounts) fail and return a `400` error with the error code `AccessControlListNotSupported`.

In contrast, a bucket with the Bucket owner preferred setting continues to accept and honor bucket and object ACLs. With this setting, new objects that are written with the `bucket-owner-full-control` canned ACL are automatically owned by the bucket owner rather than the object writer. All other ACL behaviors remain in place. To require all Amazon S3 PUT operations to include the `bucket-owner-full-control` canned ACL, you can add a bucket policy that allows only object uploads using this ACL.

To see which Object Ownership settings are applied to your buckets, you can use Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Using S3 Storage Lens to find Object Ownership settings.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Object Ownership settings

This table shows the impact that each Object Ownership setting has on ACLs, objects, object ownership, and object uploads.

Setting	Applies to	Effect on object ownership	Effect on ACLs	Uploads accepted
Bucket owner enforced (default)	All new and existing objects	Bucket owner owns every object.	ACLs are disabled and no longer affect access permissions to your bucket. Requests to set or update ACLs fail. However, requests to read ACLs are supported.	Uploads with bucket owner full control ACLs or uploads that don't specify an ACL

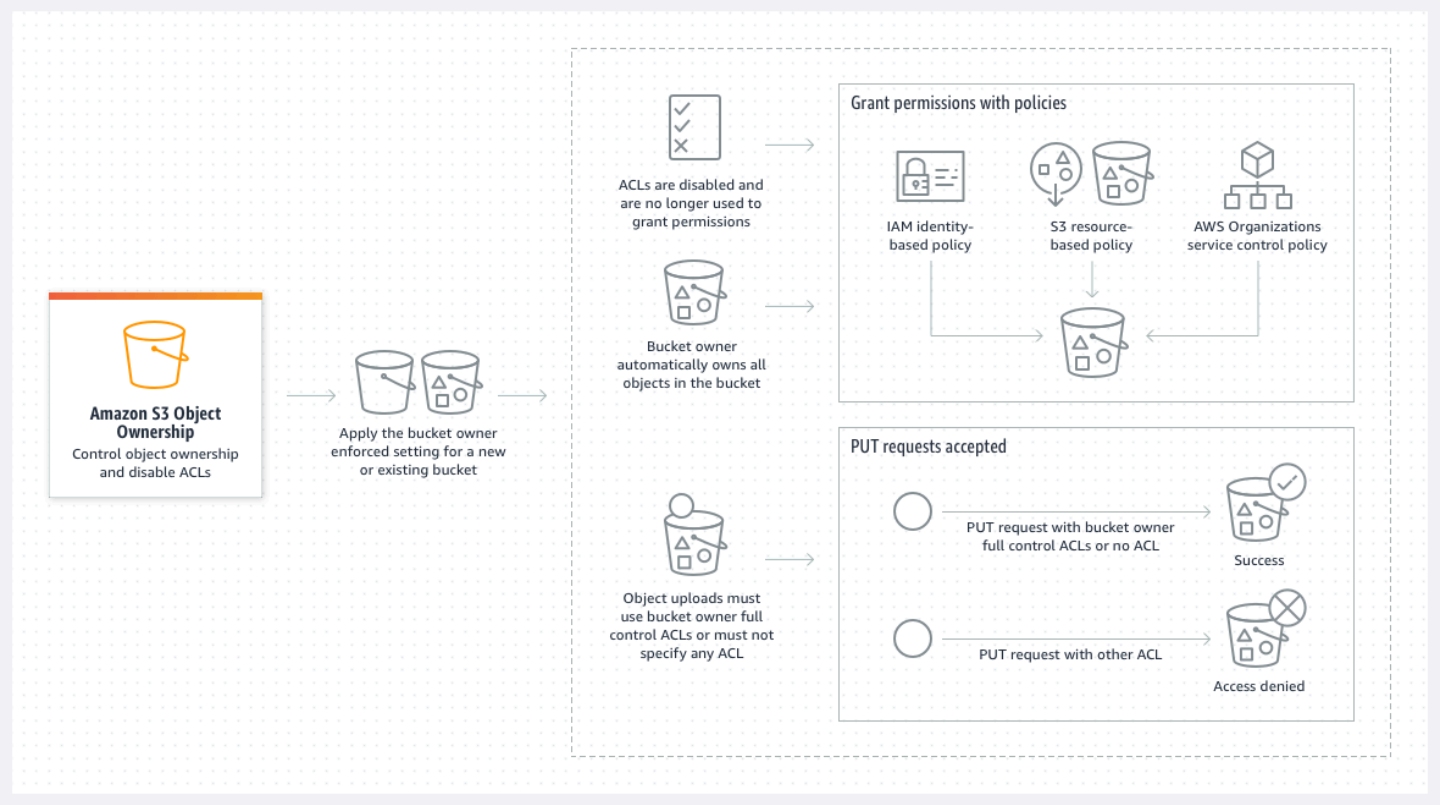
Setting	Applies to	Effect on object ownership	Effect on ACLs	Uploads accepted
			Bucket owner has full ownership and control. Object writer no longer has full ownership and control.	
Bucket owner preferred	New objects	If an object upload includes the `bucket-owner-full-control` canned ACL, the bucket owner owns the object. Objects uploaded with other ACLs are owned by the writing account.	ACLs can be updated and can grant permissions. If an object upload includes the `bucket-owner-full-control` canned ACL, the bucket owner has full control access, and the object writer no longer has full control access.	All uploads

Setting	Applies to	Effect on object ownership	Effect on ACLs	Uploads accepted
Object writer	New objects	Object writer owns the object.	ACLs can be updated and can grant permissions. Object writer has full control access.	All uploads

Changes introduced by disabling ACLs

When the Bucket owner enforced setting for Object Ownership is applied, ACLs are disabled and you automatically own and take full control over every object in the bucket without taking any additional actions. Bucket owner enforced is the default setting for all newly created buckets. After the Bucket owner enforced setting is applied, you will see three changes:

- All bucket ACLs and object ACLs are disabled, which gives full access to you, as the bucket owner. When you perform a read ACL request on your bucket or object, you will see that full access is given only to the bucket owner.
- You, as the bucket owner, automatically own and have full control over every object in your bucket.
- ACLs no longer affect access permissions to your bucket. As a result, access control for your data is based on policies, such as AWS Identity and Access Management (IAM) identity-based policies, Amazon S3 bucket policies, VPC endpoint policies, and Organizations service control policies (SCPs) or resource control policies (RCPs).



If you use S3 Versioning, the bucket owner owns and has full control over all object versions in your bucket. Applying the Bucket owner enforced setting does not add a new version of an object.

New objects can be uploaded to your bucket only if they use bucket owner full control ACLs or don't specify an ACL. Object uploads fail if they specify any other ACL. For more information, see Troubleshooting.

Because the following example `PutObject` operation using the AWS Command Line Interface (AWS CLI) includes the `bucket-owner-full-control` canned ACL, the object can be uploaded to a bucket with disabled ACLs.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key key-name --body path-to-file --
acl bucket-owner-full-control
```

Because the following `PutObject` operation doesn't specify an ACL, it also succeeds for a bucket with disabled ACLs.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key key-name --body path-to-file
```

> **Note**
>
> If other AWS accounts need access to objects after uploading, you must grant additional permissions to those accounts through bucket policies. For more information, see Walkthroughs that use policies to manage access to your Amazon S3 resources.

Re-enabling ACLs

You can re-enable ACLs by changing from the Bucket owner enforced setting to another Object Ownership setting at any time. If you used object ACLs for permissions management before you applied the Bucket owner enforced setting and you didn't migrate these object ACL permissions to your bucket policy, after you re-enable ACLs, these permissions are restored. Additionally, objects written to the bucket while the Bucket owner enforced setting was applied are still owned by the bucket owner.

For example, if you change from the Bucket owner enforced setting back to the Object writer setting, you, as the bucket owner, no longer own and have full control over objects that were previously owned by other AWS accounts. Instead, the uploading accounts again own these objects. Objects owned by other accounts use ACLs for permissions, so you can't use policies to grant permissions to these objects. However, you, as the bucket owner, still own any objects that were written to the bucket while the Bucket owner enforced setting was applied. These objects are not owned by the object writer, even if you re-enable ACLs.

For instructions on enabling and managing ACLs using the AWS Management Console, AWS Command Line Interface (CLI), REST API, or AWS SDKs, see Configuring ACLs.

Prerequisites for disabling ACLs

Before you disable ACLs for an existing bucket, complete the following prerequisites.

- Review bucket and object ACLs and migrate ACL permissions
- Identify requests that required an ACL for authorization
- Review and update bucket policies that use ACL-related condition keys

Object Ownership permissions

To apply, update, or delete an Object Ownership setting for a bucket, you need the `s3:PutBucketOwnershipControls` permission. To return the Object Ownership setting for a bucket, you need the `s3:GetBucketOwnershipControls` permission. For more information, see Setting Object Ownership when you create a bucket and Viewing the Object Ownership setting for an S3 bucket.

Disabling ACLs for all new buckets

By default, all new buckets are created with the Bucket owner enforced setting applied and ACLs are disabled. We recommend keeping ACLs disabled. As a general rule, we recommend using S3 resource-based policies (bucket policies and access point policies) or IAM policies for access control instead of ACLs. Policies are a simplified and more flexible access control option. With bucket policies and access point policies, you can define rules that apply broadly across all requests to your Amazon S3 resources.

Replication and Object Ownership

When you use S3 replication and the source and destination buckets are owned by different AWS accounts, you can disable ACLs (with the Bucket owner enforced setting for Object Ownership) to change replica ownership to the AWS account that owns the destination bucket. This setting mimics the existing owner override behavior without the need of the `s3:ObjectOwnerOverrideToBucketOwner` permission. All objects that are replicated to the destination bucket with the Bucket owner enforced setting are owned by the destination bucket owner. For more information about the owner override option for replication configurations, see Changing the replica owner.

Setting Object Ownership

You can apply an Object Ownership setting by using the Amazon S3 console, AWS CLI, AWS SDKs, Amazon S3 REST API, or AWS CloudFormation. The following REST API and AWS CLI commands support Object Ownership:

REST API	AWS CLI	Description
PutBucketOwnershipControls	put-bucket-ownership-controls	Creates or modifies the Object Ownership setting for an existing S3 bucket.
CreateBucket	create-bucket	Creates a bucket using the `x-amz-object-ownership` request header to specify the Object Ownership setting.
GetBucketOwnershipControls	get-bucket-ownership-controls	Retrieves the Object Ownership setting for an Amazon S3 bucket.
DeleteBucketOwnershipControls	delete-bucket-ownership-controls	Deletes the Object Ownership setting for an Amazon S3 bucket.

For more information about applying and working with Object Ownership settings, see the following topics.

Topics

- Prerequisites for disabling ACLs
- Setting Object Ownership when you create a bucket
- Setting Object Ownership on an existing bucket
- Viewing the Object Ownership setting for an S3 bucket
- Disabling ACLs for all new buckets and enforcing Object Ownership
- Troubleshooting

Prerequisites for disabling ACLs

A bucket access control list (ACL) in Amazon S3 is a mechanism that allows you to define granular permissions for individual objects within an S3 bucket, specifying which AWS accounts or groups can access and modify those objects. A majority of modern use cases in Amazon S3 no longer

require the use of ACLs. We recommend that you use AWS Identity and Access Management (IAM) and bucket policies to manage access, and to keep ACLs disabled, except in circumstances where you need to control access for each object individually.

If you have ACLs enabled on your bucket, before you disable ACLs, complete the following prerequisites:

Topics

- Review bucket and object ACLs and migrate ACL permissions
- Identify requests that required an ACL for authorization
- Review and update bucket policies that use ACL-related condition keys
- Example use cases

Review bucket and object ACLs and migrate ACL permissions

When you disable ACLs, permissions granted by bucket and object ACLs no longer affect access. Before you disable ACLs, review your bucket and object ACLs.

Each of your existing bucket and object ACLs has an equivalent in an IAM policy. The following bucket policy examples show you how `READ` and `WRITE` permissions for bucket and object ACLs map to IAM permissions. For more information about how each ACL translates to IAM permissions, see Mapping of ACL permissions and access policy permissions.

Before you disable ACLs:

- If your bucket ACL grants access outside of your AWS account, first, you must migrate your bucket ACL permissions to your bucket policy.
- Next, reset your bucket ACL to the default private ACL.
- We also recommend that you review your object-level ACL permissions and migrate them to your bucket policy.

If your bucket ACLs grant read or write permissions to others outside of your account, before you can disable ACLs, you must migrate these permissions to your bucket policy. After you migrate these permissions, you can set **Object Ownership** to the *Bucket owner enforced* setting. If you don't migrate bucket ACLs that grant read or write access outside of your account, your request to apply the Bucket owner enforced setting fails and returns the InvalidBucketAclWithObjectOwnership error code.

If your bucket ACL grants access outside of your AWS account, before you disable ACLs, you must migrate your bucket ACL permissions to your bucket policy and reset your bucket ACL to the default private ACL. If you don't migrate and reset, your request to apply the Bucket owner enforced setting to disable ACLs fails and returns the InvalidBucketAclWithObjectOwnership error code. We also recommend that you review your object ACL permissions and migrate them to your bucket policy.

To review and migrate ACL permissions to bucket policies, see the following topics.

Topics

- Bucket policies examples
- Using the S3 console to review and migrate ACL permissions
- Using the AWS CLI to review and migrate ACL permissions

Bucket policies examples

These example bucket policies show you how to migrate `READ` and `WRITE` bucket and object ACL permissions for a third-party AWS account to a bucket policy. `READ_ACP` and `WRITE_ACP` ACLs are less relevant for policies because they grant ACL-related permissions (`s3:GetBucketAcl`, `s3:GetObjectAcl`, `s3:PutBucketAcl`, and `s3:PutObjectAcl`).

Example– READ ACL for a bucket

If your bucket had a `READ` ACL that grants AWS account *111122223333* permission to list the contents of your bucket, you can write a bucket policy that grants `s3:ListBucket`, `s3:ListBucketVersions`, `s3:ListBucketMultipartUploads` permissions for your bucket.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
   {
    "Sid": "Permission to list the objects in a bucket",
    "Effect": "Allow",
    "Principal": {
     "AWS": [
```

```
      "arn:aws:iam::111122223333:root"
     ]
    },
    "Action": [
     "s3:ListBucket",
     "s3:ListBucketVersions",
     "s3:ListBucketMultipartUploads"
    ],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
   }
  ]
 }
```

Example– READ ACLs for every object in a bucket

If every object in your bucket has a READ ACL that grants access to AWS account *111122223333*, you can write a bucket policy that grants `s3:GetObject` and `s3:GetObjectVersion` permissions to this account for every object in your bucket.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
   {
    "Sid": "Read permission for every object in a bucket",
    "Effect": "Allow",
    "Principal": {
     "AWS": [
      "arn:aws:iam::111122223333:root"
     ]
    },
    "Action": [
     "s3:GetObject",
     "s3:GetObjectVersion"
    ],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
   }
  ]
 }
```

This example resource element grants access to a specific object.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/OBJECT-KEY"
```

Example– WRITE ACL that grants permissions to write objects to a bucket

If your bucket has a `WRITE` ACL that grants AWS account *111122223333* permission to write objects to your bucket, you can write a bucket policy that grants `s3:PutObject` permission for your bucket.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
   {
    "Sid": "Permission to write objects to a bucket",
    "Effect": "Allow",
    "Principal": {
     "AWS": [
      "arn:aws:iam::111122223333:root"
     ]
    },
    "Action": [
     "s3:PutObject"
    ],
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
   }
  ]
 }
```

Using the S3 console to review and migrate ACL permissions

To review a bucket's ACL permissions

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the bucket name.
3. Choose the **Permissions** tab.

4. Under **Access control list (ACL)**, review your bucket ACL permissions.

To review an object's ACL permissions

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the bucket name containing your object.
3. In the **Objects** list, choose your object name.
4. Choose the **Permissions** tab.
5. Under **Access control list (ACL)**, review your object ACL permissions.

To migrate ACL permissions and update your bucket ACL

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the bucket name.
3. On the **Permissions** tab, under **Bucket policy**, choose **Edit**.
4. In the **Policy** box, add or update your bucket policy.

   For example bucket policies, see Bucket policies examples and Example use cases.
5. Choose **Save changes**.
6. Update your bucket ACL to remove ACL grants to other groups or AWS accounts.
7. Apply the **Bucket owner enforced** setting for Object Ownership.

Using the AWS CLI to review and migrate ACL permissions

1. To return the bucket ACL for your bucket, use the get-bucket-acl AWS CLI command:

   ```
   aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
   ```

   For example, this bucket ACL grants `WRITE` and `READ` access to a third-party account. In this ACL, the third-party account is identified by the canonical user ID. To apply the Bucket owner enforced setting and disable ACLs, you must migrate these permissions for the third-party account to a bucket policy.

```
{
  "Owner": {
   "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
  },
  "Grants": [
   {
    "Grantee": {
     "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
     "Type": "CanonicalUser"
    },
    "Permission": "FULL_CONTROL"
   },
   {
    "Grantee": {
     "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
     "Type": "CanonicalUser"
    },
    "Permission": "READ"
   },
   {
    "Grantee": {
     "ID":
 "72806de9d1ae8b171cca9e2494a8d1335dfced4ThirdPartyAccountCanonicalUserID",
     "Type": "CanonicalUser"
    },
    "Permission": "WRITE"
   }
  ]
 }
```

For other example ACLs, see Example use cases.

2. Migrate your bucket ACL permissions to a bucket policy:

   This example bucket policy grants `s3:PutObject` and `s3:ListBucket` permissions for a third-party account. In the bucket policy, the third-party account is identified by the AWS account ID (*111122223333*).

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json
```

```
policy.json:
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "PolicyForCrossAccountAllowUpload",
   "Effect": "Allow",
   "Principal": {
    "AWS": [
     "arn:aws:iam::111122223333:root"
    ]
   },
   "Action": [
    "s3:PutObject",
    "s3:ListBucket"
   ],
   "Resource": [
    "arn:aws:s3:::amzn-s3-demo-bucket",
    "arn:aws:s3:::amzn-s3-demo-bucket/*"
  }
 ]
}
```

For more example bucket policies, see Bucket policies examples and Example use cases.

3. To return the ACL for a specific object, use the get-object-acl AWS CLI command.

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key EXAMPLE-OBJECT-KEY
```

4. If required, migrate object ACL permissions to your bucket policy.

   This example resource element grants access to a specific object in a bucket policy.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/EXAMPLE-OBJECT-KEY"
```

5. Reset the ACL for your bucket to the default ACL.

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

6. Apply the Bucket owner enforced setting for Object Ownership.

Identify requests that required an ACL for authorization

To identify Amazon S3 requests that required ACLs for authorization, you can use the `aclRequired` value in Amazon S3 server access logs or AWS CloudTrail. If the request required an ACL for authorization or if you have PUT requests that specify an ACL, the string is `Yes`. If no ACLs were required, or if you are setting a `bucket-owner-full-control` canned ACL, or if the requests are allowed by your bucket policy, the `aclRequired` value string is "`-`" in Amazon S3 server access logs and is absent in CloudTrail. For more information about the expected `aclRequired` values, see `aclRequired` values for common Amazon S3 requests.

If you have `PutBucketAcl` or `PutObjectAcl` requests with headers that grant ACL-based permissions, with the exception of the `bucket-owner-full-control` canned ACL, you must remove those headers before you can disable ACLs. Otherwise, your requests will fail.

For all other requests that required an ACL for authorization, migrate those ACL permissions to bucket policies. Then, remove any bucket ACLs before you enable the bucket owner enforced setting.

> **Note**
>
> Do not remove object ACLs. Otherwise, applications that rely on object ACLs for permissions will lose access.

If you see that no requests required an ACL for authorization, you can proceed to disable ACLs. For more information about identifying requests, see Using Amazon S3 server access logs to identify requests and Identifying Amazon S3 requests using CloudTrail.

Review and update bucket policies that use ACL-related condition keys

After you apply the Bucket owner enforced setting to disable ACLs, new objects can be uploaded to your bucket only if the request uses bucket owner full control ACLs or doesn't specify an ACL. Before disabling ACLs, review your bucket policy for ACL-related condition keys.

If your bucket policy uses an ACL-related condition key to require the `bucket-owner-full-control` canned ACL (for example, `s3:x-amz-acl`), you don't need to update your bucket policy. The following bucket policy uses the `s3:x-amz-acl` to require the `bucket-owner-full-control` canned ACL for S3 `PutObject` requests. This policy *still* requires the object writer to

specify the `bucket-owner-full-control` canned ACL. However, buckets with ACLs disabled still accept this ACL, so requests continue to succeed with no client-side changes required.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "OnlyAllowWritesToMyBucketWithBucketOwnerFullControl",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"
                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

However, if your bucket policy uses an ACL-related condition key that requires a different ACL, you must remove this condition key. This example bucket policy requires the `public-read` ACL for S3 `PutObject` requests and therefore must be updated before disabling ACLs.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Only allow writes to my bucket with public read access",
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:user/ExampleUser"                ]
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": "public-read"
                }
            }
        }
    ]
}
```

Example use cases

The following examples show you how to migrate ACL permissions to bucket policies for specific use cases.

Topics

- Grant access to the S3 log delivery group for server access logging
- Grant public read access to the objects in a bucket
- Grant Amazon ElastiCache (Redis OSS) access to your S3 bucket

Grant access to the S3 log delivery group for server access logging

If you want to apply the Bucket owner enforced setting to disable ACLs for a server access logging destination bucket (also known as a *target bucket*), you must migrate bucket ACL permissions for the S3 log delivery group to the logging service principal (`logging.s3.amazonaws.com`) in a bucket policy. For more information about log delivery permissions, see Permissions for log delivery.

This bucket ACL grants `WRITE` and `READ_ACP` access to the S3 log delivery group:

```
{
    "Owner": {
```

```
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6Bucket0wnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "Type": "CanonicalUser",
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6Bucket0wnerCanonicalUserID"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "WRITE"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

To migrate bucket ACL permissions for the S3 log delivery group to the logging service principal in a bucket policy

1. Add the following bucket policy to your destination bucket, replacing the example values.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://
policy.json

policy.json:      {
    {
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/EXAMPLE-LOGGING-PREFIX*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::SOURCE-BUCKET-NAME"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

2. Reset the ACL for your destination bucket to the default ACL.

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

3. Apply the Bucket owner enforced setting for Object Ownership to your destination bucket.

Grant public read access to the objects in a bucket

If your object ACLs grant public read access to all of the objects in your bucket, you can migrate these ACL permissions to a bucket policy.

This object ACL grants public read access to an object in a bucket:

```
{
    "Owner": {
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
```

```
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

To migrate public read ACL permissions to a bucket policy

1. To grant public read access to all of the objects in your bucket, add the following bucket policy, replacing the example values.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy
 file://policy.json

policy.json:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

   To grant public access to a specific object in a bucket policy, use the following format for the `Resource` element.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/OBJECT-KEY"
```

To grant public access to all of the objects with a specific prefix, use the following format for the `Resource` element.

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/PREFIX/*"
```

2. Apply the Bucket owner enforced setting for Object Ownership.

Grant Amazon ElastiCache (Redis OSS) access to your S3 bucket

You can export your ElastiCache (Redis OSS) backup to an S3 bucket, which gives you access to the backup from outside ElastiCache. To export your backup to an S3 bucket, you must grant ElastiCache permissions to copy a snapshot to the bucket. If you've granted permissions to ElastiCache in a bucket ACL, you must migrate these permissions to a bucket policy before you apply the Bucket owner enforced setting to disable ACLs. For more information, see Grant ElastiCache access to your Amazon S3 bucket in the *Amazon ElastiCache User Guide*.

The following example shows the bucket ACL permissions that grant permissions to ElastiCache.

```
{
    "Owner": {
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
        {
            "Grantee": {
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "READ"
```

```
        },
        {
            "Grantee": {
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "WRITE"
        },
        {
            "Grantee": {
                "ID":
 "540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353",
                "Type": "CanonicalUser"
            },
            "Permission": "READ_ACP"
        }
    ]
}
```

To migrate bucket ACL permissions for ElastiCache (Redis OSS) to a bucket policy

1. Add the following bucket policy to your bucket, replacing the example values.

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy
 file://policy.json

policy.json:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "Region.elasticache-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl",
                "s3:ListMultipartUploadParts",
```

```
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

2. Reset the ACL for your bucket to the default ACL:

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --acl private
```

3. Apply the Bucket owner enforced setting for Object Ownership.

Setting Object Ownership when you create a bucket

When you create a bucket, you can configure S3 Object Ownership. To set Object Ownership for an existing bucket, see Setting Object Ownership on an existing bucket.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to disable access control lists (ACLs) and take ownership of every object in your bucket, simplifying access management for data stored in Amazon S3. By default, S3 Object Ownership is set to the Bucket owner enforced setting, and ACLs are disabled for new buckets. With ACLs disabled, the bucket owner owns every object in the bucket and manages access to data exclusively by using access-management policies. We recommend that you keep ACLs disabled, except in unusual circumstances where you must control access for each object individually.

Object Ownership has three settings that you can use to control ownership of objects uploaded to your bucket and to disable or enable ACLs:

ACLs disabled

- **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect permissions to data in the S3 bucket. The bucket uses policies to define access control.

ACLs enabled

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

Permissions: To apply the **Bucket owner enforced** setting or the **Bucket owner preferred** setting, you must have the following permissions: `s3:CreateBucket` and `s3:PutBucketOwnershipControls`. No additional permissions are needed when creating a bucket with the **Object writer** setting applied. For more information about Amazon S3 permissions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

> **Important**
>
> A majority of modern use cases in Amazon S3 no longer require the use of ACLs, and we recommend that you disable ACLs except in circumstances where you need to control access for each object individually. With Object Ownership, you can disable ACLs and rely on policies for access control. When you disable ACLs, you can easily maintain a bucket with objects uploaded by different AWS accounts. You, as the bucket owner, own all the objects in the bucket and can manage access to them using policies.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

> **Note**
>
> - After you create a bucket, you can't change its Region.

- To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.

3. In the left navigation pane, choose **General purpose buckets**.
4. Choose **Create bucket**. The **Create bucket** page opens.
5. For **Bucket name**, enter a name for your bucket.

   The bucket name must:

   - Be unique within a partition. A partition is a grouping of Regions. AWS currently has three partitions: `aws` (commercial Regions), `aws-cn` (China Regions), and `aws-us-gov` (AWS GovCloud (US) Regions).
   - Be between 3 and 63 characters long.
   - Consist only of lowercase letters, numbers, periods (`.`), and hyphens (`-`). For best compatibility, we recommend that you avoid using periods (`.`) in bucket names, except for buckets that are used only for static website hosting.
   - Begin and end with a letter or number.
   - For a complete list of bucket-naming rules, see General purpose bucket naming rules.

   Important

   - After you create the bucket, you can't change its name.
   - Don't include sensitive information in the bucket name. The bucket name is visible in the URLs that point to the objects in the bucket.

6. (Optional) Under **General configuration**, you can choose to copy an existing bucket's settings to your new bucket. If you don't want to copy the settings of an existing bucket, skip to the next step.

   Note

   This option:

   - Isn't available in the AWS CLI and is only available in the Amazon S3 console

- Doesn't copy the bucket policy from the existing bucket to the new bucket

To copy an existing bucket's settings, under **Copy settings from existing bucket**, select **Choose bucket**. The **Choose bucket** window opens. Find the bucket with the settings that you want to copy, and select **Choose bucket**. The **Choose bucket** window closes, and the **Create bucket** window reopens.

Under **Copy settings from existing bucket**, you now see the name of the bucket that you selected. The settings of your new bucket now match the settings of the bucket that you selected. If you want to remove the copied settings, choose **Restore defaults**. Review the remaining bucket settings on the **Create bucket** page. If you don't want to make any changes, you can skip to the final step.

7. Under **Object Ownership**, to disable or enable ACLs and control ownership of objects uploaded in your bucket, choose one of the following settings:

ACLs disabled

- **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the general purpose bucket. ACLs no longer affect access permissions to data in the S3 general purpose bucket. The bucket uses policies exclusively to define access control.

  By default, ACLs are disabled. A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in circumstances where you must control access for each object individually. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

ACLs enabled

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.

  If you apply the **Bucket owner preferred** setting, to require all Amazon S3 uploads to include the `bucket-owner-full-control` canned ACL, you can add a bucket policy that allows only object uploads that use this ACL.

- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

> **Note**
>
> The default setting is **Bucket owner enforced**. To apply the default setting and keep ACLs disabled, only the `s3:CreateBucket` permission is needed. To enable ACLs, you must have the `s3:PutBucketOwnershipControls` permission.

8. Under **Block Public Access settings for this bucket**, choose the Block Public Access settings that you want to apply to the bucket.

   By default, all four Block Public Access settings are enabled. We recommend that you keep all settings enabled, unless you know that you need to turn off one or more of them for your specific use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.

   > **Note**
   >
   > To enable all Block Public Access settings, only the `s3:CreateBucket` permission is required. To turn off any Block Public Access settings, you must have the `s3:PutBucketPublicAccessBlock` permission.

9. (Optional) By default, **Bucket Versioning** is disabled. Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your bucket. With versioning, you can recover more easily from both unintended user actions and application failures. For more information about versioning, see Retaining multiple versions of objects with S3 Versioning.

   To enable versioning on your bucket, choose **Enable**.

10. (Optional) Under **Tags**, you can choose to add tags to your bucket. With AWS cost allocation, you can use bucket tags to annotate billing for your use of a bucket. A tag is a key-value pair that represents a label that you assign to a bucket. For more information, see the section called “Using cost allocation tags”.

    To add a bucket tag, enter a **Key** and optionally a **Value** and choose **Add Tag**.

11. To configure **Default encryption**, under **Encryption type**, choose one of the following:

- **Server-side encryption with Amazon S3 managed keys (SSE-S3)**
- **Server-side encryption with AWS Key Management Service keys (SSE-KMS)**
- **Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)**

> **Important**
>
> If you use the SSE-KMS or DSSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quota of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.

Buckets and new objects are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption configuration. For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets. For more information about SSE-S3, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

For more information about using server-side encryption to encrypt your data, see the section called "Data encryption".

12. If you chose **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** or **Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)**, do the following:

    a. Under **AWS KMS key**, specify your KMS key in one of the following ways:

    - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

      Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.

    - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.

- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that isn't listed, you must enter your KMS key ARN. If you want to use a KMS key that's owned by a different account, you must first have permission to use the key, and then you must enter the KMS key ARN. For more information about cross account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*. For more information about SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS). For more information about DSSE-KMS, see the section called "Dual-layer server-side encryption (DSSE-KMS)". When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

b. When you configure your bucket to use default encryption with SSE-KMS, you can also use S3 Bucket Keys. S3 Bucket Keys lower the cost of encryption by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys. S3 Bucket Keys aren't supported for DSSE-KMS.

   By default, S3 Bucket Keys are enabled in the Amazon S3 console. We recommend leaving S3 Bucket Keys enabled to lower your costs. To disable S3 Bucket Keys for your bucket, under **Bucket Key**, choose **Disable**.

13. (Optional) S3 Object Lock helps protect new objects from being deleted or overwritten. For more information, see Locking objects with Object Lock. If you want to enable S3 Object Lock, do the following:

    a. Choose **Advanced settings**.

> **Important**
>
> Enabling Object Lock automatically enables versioning for the bucket. After you've enabled and successfully created the bucket, you must also configure the Object Lock default retention and legal hold settings on the bucket's **Properties** tab.

b. If you want to enable Object Lock, choose **Enable**, read the warning that appears, and acknowledge it.

> **Note**
>
> To create an Object Lock enabled bucket, you must have the following permissions: `s3:CreateBucket`, `s3:PutBucketVersioning`, and `s3:PutBucketObjectLockConfiguration`.

14. Choose **Create bucket**.

Using the AWS CLI

To set Object Ownership when you create a new bucket, use the `create-bucket` AWS CLI command with the `--object-ownership` parameter.

This example applies the Bucket owner enforced setting for a new bucket using the AWS CLI:

```
aws s3api create-bucket --bucket  amzn-s3-demo-bucket --region us-east-1 --object-
ownership BucketOwnerEnforced
```

> **Important**
>
> If you don't set Object Ownership when you create a bucket by using the AWS CLI, the default setting will be `ObjectWriter` (ACLs enabled).

Using the AWS SDK for Java

This example sets the Bucket owner enforced setting for a new bucket using the AWS SDK for Java:

```
    // Build the ObjectOwnership for CreateBucket
```

```
    CreateBucketRequest createBucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
            .build()

     // Send the request to Amazon S3
     s3client.createBucket(createBucketRequest);
```

Using CloudFormation

To use the `AWS::S3::Bucket` CloudFormation resource to set Object Ownership when you create a new bucket, see OwnershipControls within AWS::S3::Bucket in the *AWS CloudFormation User Guide*.

Using the REST API

To apply the Bucket owner enforced setting for S3 Object Ownership, use the `CreateBucket` API operation with the `x-amz-object-ownership` request header set to `BucketOwnerEnforced`. For information and examples, see CreateBucket in the *Amazon Simple Storage Service API Reference*.

Next steps: After you apply the Bucket owner enforced or bucket owner preferred settings for Object Ownership, you can further take the following steps:

- Bucket owner enforced – Require that all new buckets are created with ACLs disabled by using an IAM or Organizations policy.
- Bucket owner preferred – Add an S3 bucket policy to require the `bucket-owner-full-control` canned ACL for all object uploads to your bucket.

Setting Object Ownership on an existing bucket

You can configure S3 Object Ownership on an existing S3 bucket. To apply Object Ownership when you create a bucket, see Setting Object Ownership when you create a bucket.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to disable access control lists (ACLs) and take ownership of every object in your bucket, simplifying access management for data stored in Amazon S3. By default, S3 Object Ownership is set to the Bucket owner enforced setting, and ACLs are disabled for new buckets. With ACLs disabled, the bucket owner owns every object in the bucket and manages access to data exclusively by using

access-management policies. We recommend that you keep ACLs disabled, except in unusual circumstances where you must control access for each object individually.

Object Ownership has three settings that you can use to control ownership of objects uploaded to your bucket and to disable or enable ACLs:

ACLs disabled

- **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect permissions to data in the S3 bucket. The bucket uses policies to define access control.

ACLs enabled

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

Prerequisites: Before you apply the Bucket owner enforced setting to disable ACLs, you must migrate bucket ACL permissions to bucket policies and reset your bucket ACLs to the default private ACL. We also recommend that you migrate object ACL permissions to bucket policies and edit bucket policies that require ACLs other than bucket owner full control ACLs. For more information, see Prerequisites for disabling ACLs.

Permissions: To use this operation, you must have the `s3:PutBucketOwnershipControls` permission. For more information about Amazon S3 permissions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to apply an S3 Object Ownership setting to.

3. Choose the **Permissions** tab.
4. Under **Object Ownership**, choose **Edit**.
5. Under **Object Ownership**, to disable or enable ACLs and control ownership of objects uploaded in your bucket, choose one of the following settings:

   ACLs disabled

   - **Bucket owner enforced** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect permissions to data in the S3 bucket. The bucket uses policies to define access control.

     To require that all new buckets are created with ACLs disabled by using IAM or AWS Organizations policies, see Disabling ACLs for all new buckets (bucket owner enforced).

   ACLs enabled

   - **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.

     If you apply the bucket owner preferred setting, to require all Amazon S3 uploads to include the `bucket-owner-full-control` canned ACL, you can add a bucket policy that only allows object uploads that use this ACL.
   - **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.
6. Choose **Save**.

Using the AWS CLI

To apply an Object Ownership setting for an existing bucket, use the `put-bucket-ownership-controls` command with the `--ownership-controls` parameter. Valid values for ownership are `BucketOwnerEnforced`, `BucketOwnerPreferred`, or `ObjectWriter`.

This example applies the Bucket owner enforced setting for an existing bucket by using the AWS CLI:

```
aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-
controls="Rules=[{ObjectOwnership=BucketOwnerEnforced}]"
```

For information about `put-bucket-ownership-controls`, see put-bucket-ownership-controls in the *AWS Command Line Interface User Guide*.

Using the AWS SDK for Java

This example applies the `BucketOwnerEnforced` setting for Object Ownership on an existing bucket by using the AWS SDK for Java:

```
        // Build the ObjectOwnership for BucketOwnerEnforced
        OwnershipControlsRule rule = OwnershipControlsRule.builder()
               .objectOwnership(ObjectOwnership.BucketOwnerEnforced)
               .build();

        OwnershipControls ownershipControls = OwnershipControls.builder()
                  .rules(rule)
                  .build()

         // Build the PutBucketOwnershipControlsRequest
         PutBucketOwnershipControlsRequest putBucketOwnershipControlsRequest =
               PutBucketOwnershipControlsRequest.builder()
                       .bucket(BUCKET_NAME)
                       .ownershipControls(ownershipControls)
                       .build();

         // Send the request to Amazon S3
         s3client.putBucketOwnershipControls(putBucketOwnershipControlsRequest);
```

Using CloudFormation

To use CloudFormation to apply an Object Ownership setting for an existing bucket, see AWS::S3::Bucket OwnershipControls in the *AWS CloudFormation User Guide*.

Using the REST API

To use the REST API to apply an Object Ownership setting to an existing S3 bucket, use `PutBucketOwnershipControls`. For more information, see PutBucketOwnershipControls in the *Amazon Simple Storage Service API Reference*.

Next steps: After you apply the Bucket owner enforced or bucket owner preferred settings for Object Ownership, you can further take the following steps:

- Bucket owner enforced – Require that all new buckets are created with ACLs disabled by using an IAM or Organizations policy.

- Bucket owner preferred – Add an S3 bucket policy to require the `bucket-owner-full-control` canned ACL for all object uploads to your bucket.

Viewing the Object Ownership setting for an S3 bucket

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to disable access control lists (ACLs) and take ownership of every object in your bucket, simplifying access management for data stored in Amazon S3. By default, S3 Object Ownership is set to the Bucket owner enforced setting, and ACLs are disabled for new buckets. With ACLs disabled, the bucket owner owns every object in the bucket and manages access to data exclusively by using access-management policies. We recommend that you keep ACLs disabled, except in unusual circumstances where you must control access for each object individually.

Object Ownership has three settings that you can use to control ownership of objects uploaded to your bucket and to disable or enable ACLs:

ACLs disabled

- **Bucket owner enforced (default)** – ACLs are disabled, and the bucket owner automatically owns and has full control over every object in the bucket. ACLs no longer affect permissions to data in the S3 bucket. The bucket uses policies to define access control.

ACLs enabled

- **Bucket owner preferred** – The bucket owner owns and has full control over new objects that other accounts write to the bucket with the `bucket-owner-full-control` canned ACL.
- **Object writer** – The AWS account that uploads an object owns the object, has full control over it, and can grant other users access to it through ACLs.

You can view the S3 Object Ownership settings for an Amazon S3 bucket. To set Object Ownership for a new bucket, see Setting Object Ownership when you create a bucket. To set Object Ownership for an existing bucket, see Setting Object Ownership on an existing bucket.

Permissions: To use this operation, you must have the `s3:GetBucketOwnershipControls` permission. For more information about Amazon S3 permissions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want to apply an Object Ownership setting to.
3. Choose the **Permissions** tab.
4. Under **Object Ownership**, you can view the Object Ownership settings for your bucket.

Using the AWS CLI

To retrieve the S3 Object Ownership setting for an S3 bucket, use the get-bucket-ownership-controls AWS CLI command.

```
aws s3api get-bucket-ownership-controls --bucket amzn-s3-demo-bucket
```

Using the REST API

To retrieve the Object Ownership setting for an S3 bucket, use the `GetBucketOwnershipControls` API operation. For more information, see GetBucketOwnershipControls.

Disabling ACLs for all new buckets and enforcing Object Ownership

We recommend that you disable ACLs on your Amazon S3 buckets. You can do this by applying the Bucket owner enforced setting for S3 Object Ownership. When you apply this setting, ACLs are disabled and you automatically own and have full control over all objects in your bucket. To require that all new buckets are created with ACLs disabled, use AWS Identity and Access Management (IAM) policies or AWS Organizations service control policies (SCPs), as described in the next section.

To enforce object ownership for new objects without disabling ACLs, you can apply the Bucket owner preferred setting. When you apply this setting, we strongly recommend that you update your bucket policy to require the `bucket-owner-full-control` canned ACL for all PUT requests to your bucket. Make sure you also update your clients to send the `bucket-owner-full-control` canned ACL to your bucket from other accounts.

Topics

- Disabling ACLs for all new buckets (bucket owner enforced)
- Requiring the bucket-owner-full-control canned ACL for Amazon S3 PUT operations (bucket owner preferred)

Disabling ACLs for all new buckets (bucket owner enforced)

The following example IAM policy denies the `s3:CreateBucket` permission for a specific IAM user or role unless the Bucket owner enforced setting is applied for Object Ownership. The key-value pair in the `Condition` block specifies `s3:x-amz-object-ownership` as its key and the `BucketOwnerEnforced` setting as its value. In other words, the IAM user can create buckets only if they set the Bucket owner enforced setting for Object Ownership and disable ACLs. You can also use this policy as a boundary SCP for your AWS organization.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "RequireBucketOwnerFullControl",
            "Action": "s3:CreateBucket",
            "Effect": "Deny",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "s3:x-amz-object-ownership": "BucketOwnerEnforced"
                }
            }
        }
    ]
}
```

Requiring the bucket-owner-full-control canned ACL for Amazon S3 PUT operations (bucket owner preferred)

With the Bucket owner preferred setting for Object Ownership, you, as the bucket owner, own and have full control over new objects that other accounts write to your bucket with the `bucket-`

owner-full-control canned ACL. However, if other accounts write objects to your bucket without the bucket-owner-full-control canned ACL, the object writer maintains full control access. You, as the bucket owner, can implement a bucket policy that allows writes only if they specify the bucket-owner-full-control canned ACL.

> **Note**
>
> If you have ACLs disabled with the Bucket owner enforced setting, you, as the bucket owner, automatically own and have full control over all the objects in your bucket. You don't need to use this section to update your bucket policy to enforce object ownership for the bucket owner.

The following bucket policy specifies that account *111122223333* can upload objects to *amzn-s3-demo-bucket* only when the object's ACL is set to bucket-owner-full-control. Be sure to replace *111122223333* with your account and *amzn-s3-demo-bucket* with the name of your bucket.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "OnlyAllowWritesToMyBucketWithBucketOwnerFullControl",
         "Effect": "Allow",
         "Principal": {
            "AWS": [
               "arn:aws:iam::111122223333:user/ExampleUser"
            ]
         },
         "Action": [
            "s3:PutObject"
         ],
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
         "Condition": {
            "StringEquals": {
               "s3:x-amz-acl": "bucket-owner-full-control"
            }
         }
```

```
        }
    ]
}
```

The following is an example copy operation that includes the `bucket-owner-full-control` canned ACL by using the AWS Command Line Interface (AWS CLI).

```
aws s3 cp file.txt s3://amzn-s3-demo-bucket --acl bucket-owner-full-control
```

After the bucket policy is put into effect, if the client does not include the `bucket-owner-full-control` canned ACL, the operation fails, and the uploader receives the following error:

An error occurred (AccessDenied) when calling the PutObject operation: Access Denied.

> **Note**
>
> If clients need access to objects after uploading, you must grant additional permissions to the uploading account. For information about granting accounts access to your resources, see Walkthroughs that use policies to manage access to your Amazon S3 resources.

Troubleshooting

When you apply the Bucket owner enforced setting for S3 Object Ownership, access control lists (ACLs) are disabled and you, as the bucket owner, automatically own all objects in your bucket. ACLs no longer affect permissions for the objects in your bucket. You can use policies to grant permissions. All S3 PUT requests must either specify the `bucket-owner-full-control` canned ACL or not specify an ACL, or these requests will fail. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

If an invalid ACL is specified or bucket ACL permissions grant access outside of your AWS account, you might see the following error responses.

AccessControlListNotSupported

After you apply the Bucket owner enforced setting for Object Ownership, ACLs are disabled. Requests to set ACLs or update ACLs fail with a `400` error and return the AccessControlListNotSupported error code. Requests to read ACLs are still supported. Requests

to read ACLs always return a response that shows full control for the bucket owner. In your PUT operations, you must either specify bucket owner full control ACLs or not specify an ACL. Otherwise, your PUT operations fail.

The following example `put-object` AWS CLI command includes the `public-read` canned ACL.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key object-key-name --body doc-example-body --acl public-read
```

If the bucket uses the Bucket owner enforced setting to disable ACLs, this operation fails, and the uploader receives the following error message:

An error occurred (AccessControlListNotSupported) when calling the PutObject operation: The bucket does not allow ACLs

InvalidBucketAclWithObjectOwnership

If you want to apply the Bucket owner enforced setting to disable ACLs, your bucket ACL must give full control only to the bucket owner. Your bucket ACL cannot give access to an external AWS account or any other group. For example, if your `CreateBucket` request sets Bucket owner enforced and specifies a bucket ACL that provides access to an external AWS account, your request fails with a `400` error and returns the InvalidBucketAclWithObjectOwnership error code. Similarly, if your `PutBucketOwnershipControls` request sets Bucket owner enforced on a bucket that has a bucket ACL that grants permissions to others, the request fails.

Example: Existing bucket ACL grants public read access

For example, if an existing bucket ACL grants public read access, you cannot apply the Bucket owner enforced setting for Object Ownership until you migrate these ACL permissions to a bucket policy and reset your bucket ACL to the default private ACL. For more information, see Prerequisites for disabling ACLs.

This example bucket ACL grants public read access:

```
{
    "Owner": {
        "ID": "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID"
    },
    "Grants": [
```

```
        {
            "Grantee": {
                "ID":
 "852b113e7a2f25102679df27bb0ae12b3f85be6BucketOwnerCanonicalUserID",
                "Type": "CanonicalUser"
            },
            "Permission": "FULL_CONTROL"
        },
        {
            "Grantee": {
                "Type": "Group",
                "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
            },
            "Permission": "READ"
        }
    ]
}
```

The following example `put-bucket-ownership-controls` AWS CLI command applies the Bucket owner enforced setting for Object Ownership:

```
aws s3api put-bucket-ownership-controls --bucket amzn-s3-demo-bucket --ownership-
controls Rules=[{ObjectOwnership=BucketOwnerEnforced}]
```

Because the bucket ACL grants public read access, the request fails and returns the following error code:

An error occurred (InvalidBucketAclWithObjectOwnership) when calling the PutBucketOwnershipControls operation: Bucket cannot have ACLs set with ObjectOwnership's BucketOwnerEnforced setting

Security in Amazon S3

Important

Amazon Simple Storage Service now applies a new default bucket security setting that automatically disables server-side encryption with customer-provided keys (SSE-C) for all new general purpose buckets. In April 2026, Amazon S3 deployed an update so all new general purpose buckets have SSE-C encryption disabled for all new write requests. For existing buckets in AWS accounts with no SSE-C encrypted objects, Amazon S3 also disabled SSE-C for all new write requests. With this change, applications that need SSE-C encryption must deliberately enable SSE-C by using the PutBucketEncryption API operation after creating a new bucket. For more information about this change, see Default SSE-C setting for new buckets FAQ.

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that are built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

Security of the cloud

AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. The effectiveness of our security is regularly tested and verified by third-party auditors as part of the AWS compliance programs. To learn about the compliance programs that apply to Amazon S3, see AWS Services in Scope by Compliance Program.

Security in the cloud

Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your organization's requirements, and applicable laws and regulations. For Amazon S3, your responsibility includes the following areas:

- Managing your data, including object ownership and encryption.
- Classifying your assets.

- Managing access to your data using IAM roles and other service configurations to apply the appropriate permissions.
- Enabling detective controls such as AWS CloudTrail or Amazon GuardDuty for Amazon S3.

This documentation will help you understand how to apply the shared responsibility model when using Amazon S3. The following topics show you how to configure Amazon S3 to meet your security and compliance objectives. You'll also learn how to use other AWS services that can help you monitor and secure your Amazon S3 resources.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Topics

- Security best practices for Amazon S3
- Data protection in Amazon S3
- Protecting data with encryption
- Internetwork traffic privacy
- Compliance validation for Amazon S3
- Resilience in Amazon S3
- Infrastructure security in Amazon S3
- Configuration and vulnerability analysis in Amazon S3
- Access management
- Amazon Simple Storage Service data inventory

Security best practices for Amazon S3

> **Important**
>
> Amazon Simple Storage Service now applies a new default bucket security setting that automatically disables server-side encryption with customer-provided keys (SSE-C) for all new general purpose buckets. In April 2026, Amazon S3 deployed an update so all new general purpose buckets have SSE-C encryption disabled for all new write requests. For existing buckets in AWS accounts with no SSE-C encrypted objects, Amazon S3 also disabled SSE-C for all new write requests. With this change, applications that need SSE-C encryption must deliberately enable SSE-C by using the PutBucketEncryption API operation after creating a new bucket. For more information about this change, see Default SSE-C setting for new buckets FAQ.

Amazon S3 provides a number of security features to consider as you develop and implement your own security policies. The following best practices are general guidelines and don't represent a complete security solution. Because these best practices might not be appropriate or sufficient for your environment, treat them as helpful recommendations rather than prescriptions.

Topics

- Amazon S3 security best practices
- Amazon S3 monitoring and auditing best practices
- Monitoring data security with managed AWS security services

Amazon S3 security best practices

The following best practices for Amazon S3 can help prevent security incidents.

Disable access control lists (ACLs)

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to control ownership of objects uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the bucket owner enforced setting and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to data exclusively using access management policies.

A majority of modern use cases in Amazon S3 no longer require the use of access control lists (ACLs). We recommend that you disable ACLs, except in circumstances where you must control access for each object individually. To disable ACLs and take ownership of every object in your bucket, apply the bucket owner enforced setting for S3 Object Ownership. When you disable ACLs, you can easily maintain a bucket with objects uploaded by different AWS accounts.

When ACLs are disabled access control for your data is based on policies, such as the following:

- AWS Identity and Access Management (IAM) user policies
- S3 bucket policies
- Virtual private cloud (VPC) endpoint policies
- AWS Organizations service control policies (SCPs)
- AWS Organizations resource control policies (RCPs)

Disabling ACLs simplifies permissions management and auditing. ACLs are disabled for new buckets by default. You can also disable ACLs for existing buckets. If you have an existing bucket that already has objects in it, after you disable ACLs, the object and bucket ACLs are no longer part of the access-evaluation process. Instead, access is granted or denied on the basis of policies.

Before you disable ACLs, make sure that you do the following:

- Review your bucket policy to ensure that it covers all the ways that you intend to grant access to your bucket outside of your account.
- Reset your bucket ACL to the default (full control to the bucket owner).

After you disable ACLs, the following behaviors occur:

- Your bucket accepts only PUT requests that do not specify an ACL or PUT requests with bucket owner full control ACLs. These ACLs include the `bucket-owner-full-control` canned ACL or equivalent forms of this ACL that are expressed in XML.
- Existing applications that support bucket owner full control ACLs see no impact.
- PUT requests that contain other ACLs (for example, custom grants to certain AWS accounts) fail and return an HTTP status code `400 (Bad Request)` with the error code `AccessControlListNotSupported`.

For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

Ensure that your Amazon S3 buckets use the correct policies and are not publicly accessible

Unless you explicitly require anyone on the internet to be able to read or write to your S3 bucket, make sure that your S3 bucket is not public. The following are some of the steps that you can take to block public access:

- Use S3 Block Public Access. With S3 Block Public Access, you can easily set up centralized controls to limit public access to your Amazon S3 resources. These centralized controls are enforced regardless of how the resources are created. For organizations managing multiple AWS accounts, you can now use organization-level enforcement through AWS Organizations to centrally manage S3 Block Public Access settings across your entire organization with a single policy configuration.

  For more information, see Blocking public access to your Amazon S3 storage.
- Identify Amazon S3 bucket policies that allow a wildcard identity such as `"Principal": "*"` (which effectively means "anyone"). Also look for policies that allow a wildcard action `"*"` (which effectively allows the user to perform any action in the Amazon S3 bucket).
- Similarly, look for Amazon S3 bucket access control lists (ACLs) that provide read, write, or full-access to "Everyone" or "Any authenticated AWS user."
- Use the `ListBuckets` API operation to scan all of your Amazon S3 buckets. Then use `GetBucketAcl`, `GetBucketWebsite`, and `GetBucketPolicy` to determine whether each bucket has compliant access controls and a compliant configuration.
- Use AWS Trusted Advisor to inspect your Amazon S3 implementation.
- Consider implementing ongoing detective controls by using the s3-bucket-public-read-prohibited and s3-bucket-public-write-prohibited managed AWS Config Rules.

For organizations with multiple AWS accounts, consider using organization-level Block Public Access management:

- Centralized policy management: Use AWS Organizations to create a single S3 Block Public Access policy that automatically applies to all member accounts or selected organizational units (OUs).
- Automatic inheritance: When you attach the policy at the root or OU level, new member accounts automatically inherit the Block Public Access settings without individual account setup.
- Simplified compliance: Organization-level policies eliminate the need to maintain complex Service Control Policies (SCPs) for Block Public Access enforcement and reduce operational overhead of managing individual account configurations.

- Audit capabilities: Use AWS CloudTrail to monitor policy attachment and enforcement across member accounts for compliance tracking.

For more information, see Identity and Access Management for Amazon S3.

Keep server-side encryption with customer-provided keys (SSE-C) disabled unless your workload requires it

Starting April 2026, Amazon S3 automatically disables SSE-C for all new general purpose buckets and existing buckets in accounts with no SSE-C encrypted objects. We recommend keeping SSE-C disabled unless your workload has a specific requirement for this encryption offering.

SSE-C requires you to provide the encryption key with every request to read or write encrypted objects. This makes it impractical to share access with other users, roles, or AWS services that operate on your data. Objects encrypted with SSE-C cannot be natively decrypted by AWS managed services. For most workloads, server-side encryption with Amazon S3 managed keys (SSE-S3) or AWS KMS keys (SSE-KMS) provides equivalent protection with greater flexibility.

If you have existing buckets where SSE-C is still enabled, review whether it's still needed and block it if not. When SSE-C is blocked, any `PutObject`, `CopyObject`, `PostObject`, Multipart Upload, or replication request that specifies SSE-C encryption is rejected with an HTTP 403 `AccessDenied` error.

To learn more about blocking SSE-C, see the section called “Blocking or unblocking SSE-C”.

Create buckets in your account regional namespace

By default, you create global general purpose buckets in the shared global namespace. After creating a general purpose bucket in the shared global namespace, that bucket name is unavailable for anyone else to create within partition. When you delete a global general purpose bucket, the bucket name becomes available again in the global namespace for anyone to re-create.

Although Amazon S3 general purpose buckets exist in a shared global namespace, you can optionally create buckets in your account regional namespace. The account regional namespace is a reserved subdivision of the global bucket namespace where only your account can create general purpose buckets. We recommend you create buckets in your account regional namespace since these buckets are unique to your account and can never be re-created by another account. For more information, see Namespaces for general purpose buckets.

We recommend that you don't delete your global general purpose buckets in the shared global namespace. All AWS Accounts now have a default bucket quota of 10,000 buckets, reducing the need to delete empty buckets from your account. If you delete a bucket in the shared global namespace, be aware that another AWS account in the same partition can use the same bucket name for a new bucket and can therefore potentially receive requests that are intended for the deleted bucket. If you want to prevent this, or if you want to continue to use the same bucket name, don't delete the bucket. We recommend that you empty the bucket and keep it. Instead of deleting the bucket, block any bucket requests as needed. For buckets no longer in active use, we recommend emptying the bucket of all objects to minimize costs while retaining the bucket itself. For more information, see Deleting a general purpose bucket.

Implement least privilege access

When granting permissions, you decide who is getting what permissions to which Amazon S3 resources. You enable specific actions that you want to allow on those resources. Therefore, we recommend that you grant only the permissions that are required to perform a task. Implementing least privilege access is fundamental in reducing security risk and the impact that could result from errors or malicious intent.

The following tools are available to implement least privilege access:

- Policy actions for Amazon S3 and Permissions Boundaries for IAM Entities
- How Amazon S3 works with IAM
- Access control list (ACL) overview

For guidance on what to consider when choosing one or more of the preceding mechanisms, see Identity and Access Management for Amazon S3.

Use IAM roles for applications and AWS services that require Amazon S3 access

In order for applications running on Amazon EC2 or other AWS services to access Amazon S3 resources, they must include valid AWS credentials in their AWS API requests. We recommend not storing AWS credentials directly in the application or Amazon EC2 instance. These are long-term credentials that are not automatically rotated and could have a significant business impact if they are compromised.

Instead, use an IAM role to manage temporary credentials for applications or services that need to access Amazon S3. When you use a role, you don't have to distribute long-term credentials (such as a username and password or access keys) to an Amazon EC2 instance or AWS service, such as AWS Lambda. The role supplies temporary permissions that applications can use when they make calls to other AWS resources.

For more information, see the following topics in the *IAM User Guide*:

- IAM Roles
- Common Scenarios for Roles: Users, Applications, and Services

Consider encryption of data at rest

You have the following options for protecting data at rest in Amazon S3:

- **Server-side encryption** – All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can set the default encryption configuration in the destination bucket.

  Amazon S3 also provides these server-side encryption options:

  - Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)
  - Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)
  - Server-side encryption with customer-provided keys (SSE-C)

  For more information, see Protecting data with server-side encryption.
- **Client-side encryption** – Encrypt data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and related tools. As with server-side encryption, client-side encryption can help reduce risk by encrypting the data with a key that is stored in a different mechanism than the mechanism that stores the data itself.

  Amazon S3 provides multiple client-side encryption options. For more information, see Protecting data by using client-side encryption.

Enforce encryption of data in transit

You can use HTTPS (TLS) to help prevent potential attackers from eavesdropping on or manipulating network traffic by using person-in-the-middle or similar attacks. We recommend allowing only encrypted connections over HTTPS (TLS) by using the aws:SecureTransport condition in your Amazon S3 bucket policies. For more information, see the example S3 bucket policy Managing access based on HTTP or HTTPS requests. In addition to denying HTTP requests, we recommend that you set Amazon CloudWatch alarms on

`tlsDetails.tlsVersion NOT EXISTS` that alert you if HTTP access attempts are made on your content. For more information on how to configure Amazon CloudWatch alarms, see Creating CloudWatch alarms for CloudTrail events: examples and CloudTrail record contents in the *AWS CloudTrail User Guide*.

> **Important**
>
> We recommend that your application not pin Amazon S3 TLS certificates as AWS doesn't support pinning of publicly-trusted certificates. S3 automatically renews certificates and renewal can happen any time before certificate expiry. Renewing a certificate generates a new public-private key pair. If you've pinned an S3 certificate which has been recently renewed with a new public key, you won't be able to connect to S3 until your application uses the new certificate.

Also consider implementing ongoing detective controls by using the s3-bucket-ssl-requests-only managed AWS Config rule.

Consider using S3 Object Lock

With S3 Object Lock, you can store objects by using a "Write Once Read Many" (WORM) model. S3 Object Lock can help prevent accidental or inappropriate deletion of data. For example, you can use S3 Object Lock to help protect your AWS CloudTrail logs.

For more information, see Locking objects with Object Lock.

Enable S3 Versioning

S3 Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your bucket. With versioning, you can easily recover from both unintended user actions and application failures.

Also consider implementing ongoing detective controls by using the s3-bucket-versioning-enabled managed AWS Config rule.

For more information, see Retaining multiple versions of objects with S3 Versioning.

Consider using S3 Cross-Region Replication

Although Amazon S3 stores your data across multiple geographically diverse Availability Zones by default, compliance requirements might dictate that you store data at even greater

distances. With S3 Cross-Region Replication (CRR), you can replicate data between distant AWS Regions to help satisfy these requirements. CRR enables automatic, asynchronous copying of objects across buckets in different AWS Regions. For more information, see Replicating objects within and across Regions.

> **Note**
>
> CRR requires both the source and destination S3 buckets to have versioning enabled.

Also consider implementing ongoing detective controls by using the s3-bucket-replication-enabled managed AWS Config rule.

Consider using VPC endpoints for Amazon S3 access

A virtual private cloud (VPC) endpoint for Amazon S3 is a logical entity within a VPC that allows connectivity only to Amazon S3. VPC endpoints can help prevent traffic from traversing the open internet.

VPC endpoints for Amazon S3 provide multiple ways to control access to your Amazon S3 data:

- You can control the requests, users, or groups that are allowed through a specific VPC endpoint by using S3 bucket policies.
- You can control which VPCs or VPC endpoints have access to your S3 buckets by using S3 bucket policies.
- You can help prevent data exfiltration by using a VPC that does not have an internet gateway.

For more information, see Controlling access from VPC endpoints with bucket policies.

Use managed AWS security services to monitor data security

Several managed AWS security services can help you identify, assess, and monitor security and compliance risks for your Amazon S3 data. These services can also help you protect your data from those risks. These services include automated detection, monitoring, and protection capabilities that are designed to scale from Amazon S3 resources for a single AWS account to resources for organizations spanning thousands of accounts.

For more information, see Monitoring data security with managed AWS security services.

Amazon S3 monitoring and auditing best practices

The following best practices for Amazon S3 can help detect potential security weaknesses and incidents.

Identify and audit all of your Amazon S3 buckets

Identification of your IT assets is a crucial aspect of governance and security. You need to have visibility of all your Amazon S3 resources to assess their security posture and take action on potential areas of weakness. To audit your resources, we recommend doing the following:

- Use Tag Editor to identify and tag security-sensitive or audit-sensitive resources, then use those tags when you need to search for these resources. For more information, see Searching for Resources to Tag in the *Tagging AWS Resources User Guide*.
- Use S3 Inventory to audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs. For more information, see Cataloging and analyzing your data with S3 Inventory.
- Create resource groups for your Amazon S3 resources. For more information, see What are resource groups? in the *AWS Resource Groups User Guide*.

Implement monitoring by using AWS monitoring tools

Monitoring is an important part of maintaining the reliability, security, availability, and performance of Amazon S3 and your AWS solutions. AWS provides several tools and services to help you monitor Amazon S3 and your other AWS services. For example, you can monitor Amazon CloudWatch metrics for Amazon S3, particularly the `PutRequests`, `GetRequests`, `4xxErrors`, and `DeleteRequests` metrics. For more information, see Monitoring metrics with Amazon CloudWatch and Logging and monitoring in Amazon S3.

For a second example, see Example: Amazon S3 Bucket Activity. This example describes how to create a CloudWatch alarm that is triggered when an Amazon S3 API call is made to PUT or `DELETE` a bucket policy, a bucket lifecycle, or a bucket replication configuration, or to PUT a bucket ACL.

Enable Amazon S3 server access logging

Server access logging provides detailed records of the requests that are made to a bucket. Server access logs can assist you in security and access audits, help you learn about your customer base, and understand your Amazon S3 bill. For instructions on enabling server access logging, see Logging requests with server access logging.

Also consider implementing ongoing detective controls by using the s3-bucket-logging-enabled AWS Config managed rule.

Use AWS CloudTrail

AWS CloudTrail provides a record of actions taken by a user, a role, or an AWS service in Amazon S3. You can use information collected by CloudTrail to determine the following:

- The request that was made to Amazon S3
- The IP address from which the request was made
- Who made the request
- When the request was made
- Additional details about the request

For example, you can identify CloudTrail entries for PUT actions that affect data access, in particular `PutBucketAcl`, `PutObjectAcl`, `PutBucketPolicy`, and `PutBucketWebsite`.

When you set up your AWS account, CloudTrail is enabled by default. You can view recent events in the CloudTrail console. To create an ongoing record of activity and events for your Amazon S3 buckets, you can create a trail in the CloudTrail console. For more information, see Logging data events in the *AWS CloudTrail User Guide*.

When you create a trail, you can configure CloudTrail to log data events. Data events are records of resource operations performed on or within a resource. In Amazon S3, data events record object-level API activity for individual buckets. CloudTrail supports a subset of Amazon S3 object-level API operations, such as `GetObject`, `DeleteObject`, and `PutObject`. For more information about how CloudTrail works with Amazon S3, see Logging Amazon S3 API calls using AWS CloudTrail. In the Amazon S3 console, you can also configure your S3 buckets to Enabling CloudTrail event logging for S3 buckets and objects.

AWS Config provides a managed rule (`cloudtrail-s3-dataevents-enabled`) that you can use to confirm that at least one CloudTrail trail is logging data events for your S3 buckets. For more information, see cloudtrail-s3-dataevents-enabled in the *AWS Config Developer Guide*.

Enable AWS Config

Several of the best practices listed in this topic suggest creating AWS Config rules. AWS Config helps you to assess, audit, and evaluate the configurations of your AWS resources. AWS Config monitors resource configurations so that you can evaluate the recorded configurations against the desired secure configurations. With AWS Config, you can do the following:

- Review changes in configurations and relationships between AWS resources
- Investigate detailed resource-configuration histories
- Determine your overall compliance against the configurations specified in your internal guidelines

Using AWS Config can help you simplify compliance auditing, security analysis, change management, and operational troubleshooting. For more information, see Setting Up AWS Config with the Console in the *AWS Config Developer Guide*. When specifying the resource types to record, ensure that you include Amazon S3 resources.

> **Important**
>
> AWS Config managed rules only supports general purpose buckets when evaluating Amazon S3 resources. AWS Config doesn't record configuration changes for directory buckets. For more information, see AWS Config Managed Rules and List of AWS Config Managed Rules in the *AWS Config Developer Guide*.

For an example of how to use AWS Config, see How to Use AWS Config to Monitor for and Respond to Amazon S3 Buckets Allowing Public Access on the *AWS Security Blog*.

Use S3 Storage Lens

S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. S3 Storage Lens also analyzes metrics to deliver contextual recommendations that you can use to optimize storage costs and apply best practices for protecting your data.

With S3 Storage Lens, you can use metrics to generate summary insights, such as finding out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use S3 Storage Lens metrics to identify cost-optimization opportunities, implement data-protection and access-management best practices, and improve the performance of application workloads.

For example, you can identify buckets that don't have S3 Lifecycle rules to abort incomplete multipart uploads that are more than 7 days old. You can also identify buckets that aren't following data-protection best practices, such as using S3 Replication or S3 Versioning. For more information, see Understanding Amazon S3 Storage Lens.

Monitor AWS security advisories

We recommend that you regularly check the security advisories posted in Trusted Advisor for your AWS account. In particular, look for warnings about Amazon S3 buckets with "open access permissions." You can do this programmatically by using describe-trusted-advisor-checks.

Further, actively monitor the primary email address that's registered to each of your AWS accounts. AWS uses this email address to contact you about emerging security issues that might affect you.

AWS operational issues with broad impact are posted on the AWS Health Dashboard - Service health. Operational issues are also posted to individual accounts through the AWS Health Dashboard. For more information, see the AWS Health documentation.

Monitoring data security with managed AWS security services

Several managed AWS security services can help you identify, assess, and monitor security and compliance risks for your Amazon S3 data. They can also help you protect your data from those risks. These services include automated detection, monitoring, and protection capabilities that are designed to scale from Amazon S3 resources for a single AWS account to resources for organizations spanning thousands of AWS accounts.

AWS detection and response services can help you identify potential security misconfigurations, threats, or unexpected behaviors, so that you can quickly respond to potentially unauthorized or malicious activity in your environment. AWS data protection services can help you monitor and protect your data, accounts, and workloads from unauthorized access. They can also help you discover sensitive data, such as personally identifiable information (PII), in your Amazon S3 data estate.

To help you identify and evaluate data security and compliance risks, managed AWS security services generate findings to notify you of potential security events or issues with your Amazon S3 data. The findings provide relevant details that you can use to investigate, assess, and act upon these risks according to your incident-response workflows and policies. You can access findings data directly by using each service. You can also send the data to other applications, services, and systems, such as your security incident and event management system (SIEM).

To monitor the security of your Amazon S3 data, consider using these managed AWS security services.

Amazon GuardDuty

Amazon GuardDuty is a threat-detection service that continuously monitors your AWS accounts and workloads for malicious activity and delivers detailed security findings for visibility and remediation.

With the S3 protection feature in GuardDuty, you can configure GuardDuty to analyze AWS CloudTrail management and data events for your Amazon S3 resources. GuardDuty then monitors those events for malicious and suspicious activity. To inform the analysis and identify potential security risks, GuardDuty uses threat-intelligence feeds and machine learning.

GuardDuty can monitor different kinds of activity for your Amazon S3 resources. For example, CloudTrail management events for Amazon S3 include bucket-level operations, such as `ListBuckets`, `DeleteBucket`, and `PutBucketReplication`. CloudTrail data events for Amazon S3 include object-level operations, such as `GetObject`, `ListObjects`, and `PutObject`. If GuardDuty detects anomalous or potentially malicious activity, it generates a finding to notify you.

For more information, see Amazon S3 Protection in Amazon GuardDuty in the *Amazon GuardDuty User Guide*.

Amazon Detective

Amazon Detective simplifies the investigative process and helps you conduct faster, more effective security investigations. Detective provides prebuilt data aggregations, summaries, and context that can help you analyze and assess the nature and extent of possible security issues.

Detective automatically extracts time-based events, such as API calls from AWS CloudTrail and Amazon VPC Flow Logs for your AWS resources. It also ingests findings generated by Amazon GuardDuty. Detective then uses machine learning, statistical analysis, and graph theory to generate visualizations that help you conduct effective security investigations more quickly.

These visualizations provide a unified, interactive view of resource behaviors and the interactions between them over time. You can explore this behavior graph to examine potentially malicious actions, such as failed login attempts or suspicious API calls. You can also see how these actions affect resources, such as S3 buckets and objects.

For more information, see the Amazon Detective Administration Guide.

IAM Access Analyzer

AWS Identity and Access Management Access Analyzer (IAM Access Analyzer) can help you identify resources that are shared with an external entity. You can also use IAM Access Analyzer to validate IAM policies against policy grammar and best practices, and generate IAM policies based on access activity in your AWS CloudTrail logs.

IAM Access Analyzer uses logic-based reasoning to analyze resource policies in your AWS environment, such as bucket policies. With IAM Access Analyzer for S3, you're alerted when an S3 bucket is configured to allow access to anyone on the internet or other AWS accounts, including accounts outside your organization. For example, IAM Access Analyzer for S3 can report that a bucket has read or write access provided through a bucket access control list (ACL), a bucket policy, a Multi-Region Access Point policy, or an access point policy. For each public or shared bucket, you receive findings that indicate the source and level of public or shared access. With these findings, you can take immediate and precise corrective action to restore bucket access to what you intended.

For more information, see Reviewing bucket access using IAM Access Analyzer for S3.

Amazon Macie

Amazon Macie is a security service that discovers sensitive data by using machine learning and pattern matching. Macie provides visibility into data security risks, and enables automated protection against those risks. With Macie, you can automate the discovery and reporting of sensitive data in your Amazon S3 data estate to gain a better understanding of the data that your organization stores in S3.

To detect sensitive data with Macie, you can use built-in criteria and techniques that are designed to detect a large and growing list of sensitive data types for many countries and regions. These sensitive data types include multiple types of personally identifiable information (PII), financial data, and credentials data. You can also use custom criteria that you define—regular expressions that define text patterns to match and, optionally, character sequences and proximity rules that refine the results.

If Macie detects sensitive data in an S3 object, Macie generates a security finding to notify you. This finding provides information about the affected object, the types and number of occurrences of the sensitive data that Macie found, and additional details to help you investigate the affected S3 bucket and object. For more information, see the Amazon Macie User Guide.

AWS Security Hub CSPM

AWS Security Hub CSPM is a security-posture management service that performs security best-practice checks, aggregates alerts and findings from multiple sources into a single format, and enables automated remediation.

Security Hub CSPM collects and provides security findings data from integrated AWS Partner Network security solutions and AWS services, including Amazon Detective, Amazon GuardDuty, IAM Access Analyzer, and Amazon Macie. It also generates its own findings by running continuous, automated security checks based on AWS best practices and supported industry standards.

Security Hub CSPM then correlates and consolidates findings across providers to help you prioritize and process the most significant findings. It also provides support for custom actions, which you can use to invoke responses or remediation actions for specific classes of findings.

With Security Hub CSPM, you can assess the security and compliance status of your Amazon S3 resources, and you can do so as part of a broader analysis of your organization's security posture in individual AWS Regions and across multiple Regions. This includes analyzing security trends and identifying the highest-priority security issues. You can also aggregate findings from multiple AWS Regions, and monitor and process aggregated findings data from a single Region.

For more information, see Amazon Simple Storage Service controls in the *AWS Security Hub CSPM User Guide*.

Data protection in Amazon S3

Amazon S3 provides a highly durable storage infrastructure designed for mission-critical and primary data storage. S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive redundantly store objects on multiple devices across a minimum of three Availability Zones in an AWS Region. An Availability Zone is one or more discrete data centers with redundant power, networking, and connectivity in an AWS Region. Availability Zones are physically separated by a meaningful distance, many kilometers, from any other Availability Zone, although all are within 100 km (60 miles) of each other. The S3 One Zone-IA storage class stores data redundantly across multiple devices within a single Availability Zone. These services are designed to handle concurrent device failures by quickly detecting and repairing any lost redundancy, and they also regularly verify the integrity of your data using checksums.

Amazon S3 standard storage offers the following features:

- Backed with the Amazon S3 Service Level Agreement.
- Designed to provide 99.999999999% durability and 99.99% availability of objects over a given year.
- S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive are all designed to sustain data in the event of the loss of an entire Amazon S3 Availability Zone.

Amazon S3 further protects your data using versioning. You can use versioning to preserve, retrieve, and restore every version of every object that is stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. By default, requests retrieve the most recently written version. You can retrieve older versions of an object by specifying a version of the object in a request.

Apart from S3 Versioning, you can also use Amazon S3 Object Lock and S3 Replication to protect your data. For more information, see Data protection in Amazon S3.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual user accounts with AWS Identity and Access Management, so that each user is given only the permissions necessary to fulfill their job duties.

If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see Federal Information Processing Standard (FIPS) 140-2.

The following security best practices also address data protection in Amazon S3:

- Implement server-side encryption
- Enforce encryption of data in transit
- Consider using Macie with Amazon S3
- Identify and audit all your Amazon S3 buckets
- Monitor Amazon Web Services security advisories

Protecting data with encryption

Important

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

Data protection refers to protecting data while it's in transit (as it travels to and from Amazon S3) and at rest (while it is stored on disks in Amazon S3 data centers). You can protect data in transit by using Secure Socket Layer/Transport Layer Security (SSL/TLS) including hybrid post-quantum key exchange or client-side encryption. For protecting data at rest in Amazon S3, you have the following options:

- **Server-side encryption** – Amazon S3 encrypts your objects before saving them on disks in AWS data centers and then decrypts the objects when you download them.

  All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can update the default encryption configuration in the destination bucket.

  If you want to specify a different encryption type in your PUT requests, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If you want to set a different default encryption configuration in the destination bucket, you can use SSE-KMS or DSSE-KMS.

  For more information about changing the default encryption configuration for your general purpose buckets, see Configuring default encryption.

When you change the default encryption configuration of your bucket to SSE-KMS, the encryption type of the existing Amazon S3 objects in the bucket is not changed. To change the encryption type of your pre-existing objects after updating the default encryption configuration to SSE-KMS, you can use Amazon S3 Batch Operations. You provide S3 Batch Operations with a list of objects, and Batch Operations calls the respective API operation. You can use the Copy objects action to copy existing objects, which writes them back to the same bucket as SSE-KMS encrypted objects. A single Batch Operations job can perform the specified operation on billions of objects. For more information, see Performing object operations in bulk with Batch Operations and the *AWS Storage Blog* post How to retroactively encrypt existing objects in Amazon S3 using S3 Inventory, Amazon Athena, and S3 Batch Operations.

For more information about each option for server-side encryption, see Protecting data with server-side encryption.

To configure server-side encryption, see:

- Specifying server-side encryption with Amazon S3 managed keys (SSE-S3)
- Specifying server-side encryption with AWS KMS (SSE-KMS)
- the section called "Specifying DSSE-KMS"
- Specifying server-side encryption with customer-provided keys (SSE-C)

- **Client-side encryption** – You encrypt your data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, encryption keys, and related tools.

  To configure client-side encryption, see Protecting data by using client-side encryption.

To see which percentage of your storage bytes are encrypted, you can use Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Assessing your storage activity and usage with S3 Storage Lens. For a complete list of metrics, see S3 Storage Lens metrics glossary.

For more information about server-side encryption, client-side encryption, and encryption in transit, review the following topics.

Topics

- Protecting data with server-side encryption

- Protecting data by using client-side encryption
- Protecting data in transit with encryption

Protecting data with server-side encryption

Important

Amazon Simple Storage Service now applies a new default bucket security setting that automatically disables server-side encryption with customer-provided keys (SSE-C) for all new general purpose buckets. In April 2026, Amazon S3 deployed an update so all new general purpose buckets have SSE-C encryption disabled for all new write requests. For existing buckets in AWS accounts with no SSE-C encrypted objects, Amazon S3 also disabled SSE-C for all new write requests. With this change, applications that need SSE-C encryption must deliberately enable SSE-C by using the PutBucketEncryption API operation after creating a new bucket. For more information about this change, see Default SSE-C setting for new buckets FAQ.

Important

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

Server-side encryption is the encryption of data at its destination by the application or service that receives it. Amazon S3 encrypts your data at the object level as it writes it to disks in AWS data centers and decrypts it for you when you access it. As long as you authenticate your request and you have access permissions, there is no difference in the way you access encrypted or unencrypted objects. For example, if you share your objects by using a presigned URL, that URL works the

same way for both encrypted and unencrypted objects. Additionally, when you list objects in your bucket, the list API operations return a list of all objects, regardless of whether they are encrypted.

All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can update the default encryption configuration in the destination bucket.

If you want to specify a different encryption type in your PUT requests, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If you want to set a different default encryption configuration in the destination bucket, you can use SSE-KMS or DSSE-KMS.

For more information about changing the default encryption configuration for your general purpose buckets, see Configuring default encryption.

When you change the default encryption configuration of your bucket to SSE-KMS, the encryption type of the existing Amazon S3 objects in the bucket is not changed. To change the encryption type of your pre-existing objects after updating the default encryption configuration to SSE-KMS, you can use Amazon S3 Batch Operations. You provide S3 Batch Operations with a list of objects, and Batch Operations calls the respective API operation. You can use the Copy objects action to copy existing objects, which writes them back to the same bucket as SSE-KMS encrypted objects. A single Batch Operations job can perform the specified operation on billions of objects. For more information, see Performing object operations in bulk with Batch Operations and the *AWS Storage Blog* post How to retroactively encrypt existing objects in Amazon S3 using S3 Inventory, Amazon Athena, and S3 Batch Operations.

> **Note**
>
> You can't apply different types of server-side encryption to the same object simultaneously.

If you need to encrypt your existing objects, use S3 Batch Operations and S3 Inventory. For more information, see Encrypting objects with Amazon S3 Batch Operations and Performing object operations in bulk with Batch Operations.

When storing data in Amazon S3 you have four mutually exclusive options for server-side encryption, depending on how you choose to manage the encryption keys and the number of encryption layers that you want to apply.

Server-side encryption with Amazon S3 managed keys (SSE-S3)

All Amazon S3 buckets have encryption configured by default. The default option for server-side encryption is with Amazon S3 managed keys (SSE-S3). Each object is encrypted with a unique key. As an additional safeguard, SSE-S3 encrypts the key itself with a root key that it regularly rotates. SSE-S3 uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data. For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)

Server-side encryption with AWS KMS keys (SSE-KMS) is provided through an integration of the AWS KMS service with Amazon S3. With AWS KMS, you have more control over your keys. For example, you can view separate keys, edit control policies, and follow the keys in AWS CloudTrail. Additionally, you can create and manage customer managed keys or use AWS managed keys that are unique to you, your service, and your Region. For more information, see Using server-side encryption with AWS KMS keys (SSE-KMS).

Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS)

Dual-layer server-side encryption with AWS KMS keys (DSSE-KMS) is similar to SSE-KMS, but DSSE-KMS applies two independent layers of AES-256 encryption instead of one layer: first using a AWS KMS data encryption key, then using a separate Amazon S3-managed encryption key. Because both layers of encryption are applied to an object on the server side, you can use a wide range of AWS services and tools to analyze data in S3 while using an encryption method that can satisfy compliance requirements for multilayer encryption. For more information, see Using dual-layer server-side encryption with AWS KMS keys (DSSE-KMS).

Server-side encryption with customer-provided keys (SSE-C)

With server-side encryption with customer-provided keys (SSE-C), you manage the encryption keys, and Amazon S3 manages the encryption as it writes to disks and the decryption when you access your objects. For more information, see Using server-side encryption with customer-provided keys (SSE-C).

Note

Objects encrypted with SSE-C do not support annotations. Attempting to add an annotation to an SSE-C encrypted object returns an error. For all other encryption types, annotations inherit the encryption configuration of the parent object. S3 Bucket Keys are supported for annotations on SSE-KMS encrypted objects.

Note

When using access points for Amazon FSx file systems using S3 access points you have one option for server-side encryption.

Server-side encryption with Amazon FSx (SSE-FSX)

All Amazon FSx file systems have encryption configured by default and are encrypted at rest with keys managed using AWS Key Management Service. Data is automatically encrypted and decrypted by on the file system as data is being written to and read from the file system. These processes are handled transparently by Amazon FSx.

Setting default server-side encryption behavior for Amazon S3 buckets

Important

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

All Amazon S3 buckets have encryption configured by default, and objects are automatically encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3). This encryption setting applies to all objects in your Amazon S3 buckets.

If you need more control over your keys, such as managing key rotation and access policy grants, you can choose to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS). For more information about editing KMS keys, see Editing keys in *AWS Key Management Service Developer Guide*.

> **Note**
>
> We've changed buckets to encrypt new object uploads automatically. If you previously created a bucket without default encryption, Amazon S3 will enable encryption by default for the bucket using SSE-S3. There will be no changes to the default encryption configuration for an existing bucket that already has SSE-S3 or SSE-KMS configured. If you want to encrypt your objects with SSE-KMS, you must change the encryption type in your bucket settings. For more information, see Using server-side encryption with AWS KMS keys (SSE-KMS).

When you configure your bucket to use default encryption with SSE-KMS, you can also enable S3 Bucket Keys to decrease request traffic from Amazon S3 to AWS KMS and reduce the cost of encryption. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

To identify buckets that have SSE-KMS enabled for default encryption, you can use Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Using S3 Storage Lens to protect your data.

When you use server-side encryption, Amazon S3 encrypts an object before saving it to disk and decrypts it when you download the object. For more information about protecting data using server-side encryption and encryption-key management, see Protecting data with server-side encryption.

For more information about the permissions required for default encryption, see PutBucketEncryption in the *Amazon Simple Storage Service API Reference*.

You can configure the Amazon S3 default encryption behavior for an S3 bucket by using the Amazon S3 console, the AWS SDKs, the Amazon S3 REST API, and the AWS Command Line Interface (AWS CLI).

Encrypting existing objects

To encrypt your existing unencrypted Amazon S3 objects, you can use Amazon S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on, and Batch Operations calls the respective API to perform the specified operation. You can use the Batch Operations Copy operation to copy existing unencrypted objects and write them back to the same bucket as encrypted objects. A single Batch Operations job can perform the specified operation on billions of objects. For more information, see Performing object operations in bulk with Batch Operations and the *AWS Storage Blog* post Encrypting objects with Amazon S3 Batch Operations.

You can also encrypt existing objects by using the `CopyObject` API operation or the `copy-object` AWS CLI command. For more information, see the *AWS Storage Blog* post Encrypting existing Amazon S3 objects with the AWS CLI.

> **Important**
>
> If your bucket is a target destination for server access log delivery, the bucket must use Amazon S3 managed keys (SSE-S3). If the destination bucket uses SSE-KMS default encryption, log objects might be created but encrypted with a key that you can't access.

Using SSE-KMS encryption for cross-account operations

When using encryption for cross-account operations, be aware of the following:

- If an AWS KMS key Amazon Resource Name (ARN) or alias is not provided at request time or through the bucket's default encryption configuration, the AWS managed key (`aws/s3`) is used.
- If you're uploading or accessing S3 objects by using AWS Identity and Access Management (IAM) principals that are in the same AWS account as your KMS key, you can use the AWS managed key (`aws/s3`).
- If you want to grant cross-account access to your S3 objects, use a customer managed key. You can configure the policy of a customer managed key to allow access from another account.
- If you're specifying a customer managed KMS key, we recommend using a fully qualified KMS key ARN. If you use a KMS key alias instead, AWS KMS resolves the key within the requester's account. This behavior can result in data that's encrypted with a KMS key that belongs to the requester, and not the bucket owner.
- You must specify a key that you (the requester) have been granted `Encrypt` permission to. For more information, see Allow key users to use a KMS key for cryptographic operations in the *AWS Key Management Service Developer Guide*.

For more information about when to use customer managed keys and AWS managed KMS keys, see Should I use an AWS managed key or a customer managed key to encrypt my objects in Amazon S3?

Using default encryption with replication

When you enable default encryption for a replication destination bucket, the following encryption behavior applies:

- If objects in the source bucket are not encrypted, the replica objects in the destination bucket are encrypted by using the default encryption settings of the destination bucket. As a result, the entity tags (ETags) of the source objects differ from the ETags of the replica objects. If you have applications that use ETags, you must update those applications to account for this difference.
- If objects in the source bucket are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), the replica objects in the destination bucket use the same type of encryption as the source objects. The default encryption settings of the destination bucket are not used.

For more information about using default encryption with SSE-KMS, see Replicating encrypted objects.

Using Amazon S3 Bucket Keys with default encryption

When you configure your bucket to use SSE-KMS as the default encryption behavior for new objects, you can also configure S3 Bucket Keys. S3 Bucket Keys decrease the number of transactions from Amazon S3 to AWS KMS to reduce the cost of SSE-KMS.

When you configure your bucket to use S3 Bucket Keys for SSE-KMS on new objects, AWS KMS generates a bucket-level key that is used to create a unique data key for objects in the bucket. This S3 Bucket Key is used for a time-limited period within Amazon S3, reducing the need for Amazon S3 to make requests to AWS KMS to complete encryption operations.

For more information about using S3 Bucket Keys, see Using Amazon S3 Bucket Keys.

Configuring default encryption

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

Amazon S3 buckets have bucket encryption enabled by default, and new objects are automatically encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3). This encryption applies to all new objects in your Amazon S3 buckets, and comes at no cost to you.

If you need more control over your encryption keys, such as managing key rotation and access policy grants, you can elect to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS). For more information about SSE-KMS, see Specifying server-side encryption with AWS KMS (SSE-KMS). For more information about DSSE-KMS, see the section called "Dual-layer server-side encryption (DSSE-KMS)".

If you want to use a KMS key that is owned by a different account, you must have permission to use the key. For more information about cross-account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*.

When you set default bucket encryption to SSE-KMS, you can also configure an S3 Bucket Key to reduce your AWS KMS request costs. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

> **Note**
>
> If you use PutBucketEncryption to set your default bucket encryption to SSE-KMS, you should verify that your KMS key ID is correct. Amazon S3 does not validate the KMS key ID provided in PutBucketEncryption requests.

There are no additional charges for using default encryption for S3 buckets. Requests to configure the default encryption behavior incur standard Amazon S3 request charges. For information about pricing, see Amazon S3 pricing. For SSE-KMS and DSSE-KMS, AWS KMS charges apply and are listed at AWS KMS pricing.

Server-side encryption with customer-provided keys (SSE-C) is not supported for default encryption.

You can configure Amazon S3 default encryption for an S3 bucket by using the Amazon S3 console, the AWS SDKs, the Amazon S3 REST API, and the AWS Command Line Interface (AWS CLI).

Changes to note before enabling default encryption

After you enable default encryption for a bucket, the following encryption behavior applies:

- There is no change to the encryption of the objects that existed in the bucket before default encryption was enabled.
- When you upload objects after enabling default encryption:
  - If your PUT request headers don't include encryption information, Amazon S3 uses the bucket's default encryption settings to encrypt the objects.
  - If your PUT request headers include encryption information, Amazon S3 uses the encryption information from the PUT request to encrypt objects before storing them in Amazon S3.
- If you use the SSE-KMS or DSSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quotas of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.
- If this bucket is used as a destination for server access logging, the destination bucket must use Amazon S3 managed keys (SSE-S3). If the destination bucket uses SSE-KMS default encryption, Amazon S3 might deliver log objects that are encrypted with a key that you can't access. To resolve this, change the destination bucket's default encryption to SSE-S3. For more information about server access logging, see Logging requests with server access logging.

> **Note**
>
> Objects uploaded before default encryption was enabled will not be encrypted. For information about encrypting existing objects, see the section called "Setting default bucket encryption".

Using the S3 console

To configure default encryption on an Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want.
4. Choose the **Properties** tab.
5. Under **Default encryption**, choose **Edit**.
6. To configure encryption, under **Encryption type**, choose one of the following:

   - **Server-side encryption with Amazon S3 managed keys (SSE-S3)**
   - **Server-side encryption with AWS Key Management Service keys (SSE-KMS)**
   - **Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)**

   > **Important**
   >
   > If you use the SSE-KMS or DSSE-KMS options for your default encryption configuration, you are subject to the requests per second (RPS) quotas of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.

   Buckets and new objects are encrypted by default with SSE-S3, unless you specify another type of default encryption for your buckets. For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets.

   For more information about using Amazon S3 server-side encryption to encrypt your data, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).
7. If you chose **Server-side encryption with AWS Key Management Service keys (SSE-KMS)** or **Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)**, do the following:

   a. Under **AWS KMS key**, specify your KMS key in one of the following ways:

- To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

  Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
- To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can only use KMS keys that are enabled in the same AWS Region as the bucket. When you choose **Choose from your KMS keys**, the S3 console only lists 100 KMS keys per Region. If you have more than 100 KMS keys in the same Region, you can only see the first 100 KMS keys in the S3 console. To use a KMS key that is not listed in the console, choose **Enter AWS KMS key ARN**, and enter the KMS key ARN.
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 only supports symmetric encryption KMS keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

For more information about using SSE-KMS with Amazon S3, see Using server-side encryption with AWS KMS keys (SSE-KMS). For more information about using DSSE-KMS, see the section called "Dual-layer server-side encryption (DSSE-KMS)".

b. When you configure your bucket to use default encryption with SSE-KMS, you can also enable an S3 Bucket Key. S3 Bucket Keys lower the cost of encryption by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

To use S3 Bucket Keys, under **Bucket Key**, choose **Enable**.

> **Note**
>
> S3 Bucket Keys aren't supported for DSSE-KMS.

8. Choose **Save changes**.

Using the AWS CLI

These examples show you how to configure default encryption by using SSE-S3 or by using SSE-KMS with an S3 Bucket Key.

For more information about default encryption, see Setting default server-side encryption behavior for Amazon S3 buckets. For more information about using the AWS CLI to configure default encryption, see put-bucket-encryption.

Example– Default encryption with SSE-S3

This example configures default bucket encryption with Amazon S3 managed keys.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
    "Rules": [
        {
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }
    ]
}'
```

Example– Default encryption with SSE-KMS using an S3 Bucket Key

This example configures default bucket encryption with SSE-KMS using an S3 Bucket Key.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
    "Rules": [
```

```
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

Using the REST API

Use the REST API `PutBucketEncryption` operation to enable default encryption and to set the type of server-side encryption to use—SSE-S3, SSE-KMS, or DSSE-KMS.

For more information, see PutBucketEncryption in the *Amazon Simple Storage Service API Reference*.

Monitoring default encryption with AWS CloudTrail and Amazon EventBridge

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

You can track default encryption configuration requests for Amazon S3 buckets by using AWS CloudTrail events. The following API event names are used in CloudTrail logs:

- `PutBucketEncryption`
- `GetBucketEncryption`
- `DeleteBucketEncryption`

You can also create EventBridge rules to match the CloudTrail events for these API calls. For more information about CloudTrail events, see Enable logging for objects in a bucket using the console. For more information about EventBridge events, see Events from AWS services.

You can use CloudTrail logs for object-level Amazon S3 actions to track PUT and POST requests to Amazon S3. You can use these actions to verify whether default encryption is being used to encrypt objects when incoming PUT requests don't have encryption headers.

When Amazon S3 encrypts an object by using the default encryption settings, the log includes one of the following fields as the name-value pair: `"SSEApplied":"Default_SSE_S3"`, `"SSEApplied":"Default_SSE_KMS"`, or `"SSEApplied":"Default_DSSE_KMS"`.

When Amazon S3 encrypts an object by using the PUT encryption headers, the log includes one of the following fields as the name-value pair: `"SSEApplied":"SSE_S3"`, `"SSEApplied":"SSE_KMS"`, `"SSEApplied":"DSSE_KMS"`, or `"SSEApplied":"SSE_C"`.

For multipart uploads, this information is included in your `InitiateMultipartUpload` API operation requests. For more information about using CloudTrail and CloudWatch, see Logging and monitoring in Amazon S3.

Default encryption FAQ

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. SSE-S3, which uses 256-bit Advanced Encryption Standard (AES-256), is automatically applied to all new buckets and to any existing S3 bucket that doesn't already have default encryption configured. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in AWS CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface (AWS CLI) and the AWS SDKs.

The following sections answer questions about this update.

Does Amazon S3 change the default encryption settings for my existing buckets that already have default encryption configured?

No. There are no changes to the default encryption configuration for an existing bucket that already has SSE-S3 or server-side encryption with AWS Key Management Service (AWS KMS) keys

(SSE-KMS) configured. For more information about how to set the default encryption behavior for buckets, see Setting default server-side encryption behavior for Amazon S3 buckets. For more information about SSE-S3 and SSE-KMS encryption settings, see Protecting data with server-side encryption.

Is default encryption enabled on my existing buckets that don't have default encryption configured?

Yes. Amazon S3 now configures default encryption on all existing unencrypted buckets to apply server-side encryption with S3 managed keys (SSE-S3) as the base level of encryption for new objects uploaded to these buckets. Objects that are already in an existing unencrypted bucket won't be automatically encrypted.

How can I view the default encryption status of new object uploads?

Currently, you can view the default encryption status of new object uploads in AWS CloudTrail logs, S3 Inventory, and S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS Command Line Interface (AWS CLI) and the AWS SDKs.

- To view your CloudTrail events, see Viewing CloudTrail events in the CloudTrail console in the *AWS CloudTrail User Guide*. CloudTrail logs provide API tracking for PUT and POST requests to Amazon S3. When default encryption is being used to encrypt objects in your buckets, the CloudTrail logs for PUT and POST API requests will include the following field as the name-value pair: `"SSEApplied":"Default_SSE_S3"`.
- To view the automatic encryption status of new object uploads in S3 Inventory, configure an S3 Inventory report to include the **Encryption** metadata field, and then see the encryption status of each new object in the report. For more information, see Setting up Amazon S3 Inventory.
- To view the automatic encryption status for new object uploads in S3 Storage Lens, configure an S3 Storage Lens dashboard and see the **Encrypted bytes** and **Encrypted object count** metrics in the **Data protection** category of the dashboard. For more information, see Using the S3 console and Viewing S3 Storage Lens metrics on the dashboards.
- To view the automatic bucket-level encryption status in the Amazon S3 console, check the **Default encryption** of your Amazon S3 buckets in the Amazon S3 console. For more information, see Configuring default encryption.
- To view the automatic encryption status as an additional Amazon S3 API response header in the AWS Command Line Interface (AWS CLI) and the AWS SDKs, check the response header `x-amz-server-side-encryption` when you use object action APIs, such as PutObject and GetObject.

What do I have to do to take advantage of this change?

You are not required to make any changes to your existing applications. Because default encryption is enabled for all of your buckets, all new objects uploaded to Amazon S3 are automatically encrypted.

Can I disable encryption for the new objects being written to my bucket?

No. SSE-S3 is the new base level of encryption that's applied to all the new objects being uploaded to your bucket. You can no longer disable encryption for new object uploads.

Will my charges be affected?

No. Default encryption with SSE-S3 is available at no additional cost. You will be billed for storage, requests, and other S3 features, as usual. For pricing, see Amazon S3 pricing.

Will Amazon S3 encrypt my existing objects that are unencrypted?

No. Beginning on January 5, 2023, Amazon S3 only automatically encrypts new object uploads. To encrypt existing objects, you can use S3 Batch Operations to create encrypted copies of your objects. These encrypted copies will retain the existing object data and name and will be encrypted by using the encryption keys that you specify. For more details, see Encrypting objects with Amazon S3 Batch Operations in the *AWS Storage Blog*.

I did not enable encryption for my buckets before this release. Do I need to change the way that I access objects?

No. Default encryption with SSE-S3 automatically encrypts your data as it's written to Amazon S3 and decrypts it for you when you access it. There is no change in the way that you access objects that are automatically encrypted.

Do I need to change the way that I access my client-side encrypted objects?

No. All client-side encrypted objects that are encrypted before being uploaded into Amazon S3 arrive as encrypted ciphertext objects within Amazon S3. These objects will now have an additional layer of SSE-S3 encryption. Your workloads that use client-side encrypted objects will not require any changes to your client services or authorization settings.

> **Note**
>
> HashiCorp Terraform users that aren't using an updated version of the AWS Provider might see an unexpected drift after creating new S3 buckets with no customer defined encryption

configuration. To avoid this drift, update your Terraform AWS Provider version to one of the following versions: any 4.x release, 3.76.1, or 2.70.4.

Updating server-side encryption for existing data

All Amazon S3 buckets have encryption configured by default, and objects are automatically encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3). This default encryption setting applies to all new objects in your Amazon S3 buckets.

Using the `UpdateObjectEncryption` API operation, you can atomically update the server-side encryption type of an existing encrypted object in a general purpose bucket from server-side encryption with Amazon S3 managed encryption (SSE-S3) to server-side encryption with AWS Key Management Service (AWS KMS) encryption keys (SSE-KMS). The `UpdateObjectEncryption` API operation uses envelope encryption to re-encrypt the data key used to encrypt and decrypt your object with your newly specified server-side encryption type.

Amazon S3 performs this encryption type update without any data movement. In other words, when you use the `UpdateObjectEncryption` operation, your data isn't copied, archived objects in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive aren't restored, and objects in the S3 Intelligent-Tiering storage class aren't moved between tiers. Additionally, the `UpdateObjectEncryption` operation preserves all object metadata properties, including the storage class, creation date, last modified date, ETag, and checksum properties.

The `UpdateObjectEncryption` operation is supported for all S3 storage classes that are supported by general purpose buckets. You can use the `UpdateObjectEncryption` operation to do the following:

- Change encrypted objects from server-side encryption with Amazon S3 managed encryption (SSE-S3) to server-side encryption with AWS Key Management Service (AWS KMS) encryption keys (SSE-KMS).
- Update object-level SSE-KMS encrypted objects to use S3 Bucket Keys, which decreases the AWS KMS request traffic from Amazon S3 to AWS KMS. For more information, see the section called "Using Amazon S3 Bucket Keys".
- Change the customer-managed KMS key that's used to encrypt your data so that you can comply with custom key-rotation standards.

> **Note**
>
> Source objects that are unencrypted, or encrypted with either dual-layer server-side encryption with AWS KMS keys (DSSE-KMS) or customer-provided encryption keys (SSE-C) aren't supported by this operation.

The `UpdateObjectEncryption` operation is typically completed in milliseconds regardless of the size of the object or the storage class, including S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive. This operation doesn't count as an access for S3 Intelligent-Tiering, so objects in the Infrequent Access tier or the Archive Instant Access tier won't automatically tier back to the Frequent Access tier if you change the server-side encryption type of your object.

`UpdateObjectEncryption` is an object-level (data plane) API operation that's logged to Amazon S3 server access logs and AWS CloudTrail data events. For more information, see the section called "Logging options".

The `UpdateObjectEncryption` operation is priced the same as `PUT`, `COPY`, `POST`, and `LIST` requests (per 1,000 requests) and is always charged as an S3 Standard storage class request regardless of the underlying object's storage class. For more information, see Amazon S3 pricing.

Restrictions and considerations

When using the `UpdateObjectEncryption` operation, the following restrictions and considerations apply:

- The `UpdateObjectEncryption` operation doesn't support objects that are unencrypted or objects that are encrypted with either dual-layer server-side encryption with AWS KMS keys (DSSE-KMS) or customer-provided encryption keys (SSE-C). Additionally, you cannot specify SSE-S3 as the requested new encryption type `UpdateObjectEncryption` request.
- You can use the `UpdateObjectEncryption` operation to update objects in buckets that have S3 Versioning enabled. To update the encryption type of a particular version, you must specify a version ID in your `UpdateObjectEncryption` request. If you don't specify version ID, the `UpdateObjectEncryption` request acts on the current version of the object. For more information about S3 Versioning, see the section called "Retaining multiple versions of objects".
- The `UpdateObjectEncryption` operation fails on any object that has an S3 Object Lock retention mode or legal hold applied to it. If an object has a governance-mode retention period or a legal hold, you must first remove the Object Lock status on the object before you issue your

`UpdateObjectEncryption` request. You can't use the `UpdateObjectEncryption` operation with objects that have an Object Lock compliance mode retention period applied to them. For more information about S3 Object Lock, see the section called “Locking objects”.

- `UpdateObjectEncryption` requests on source buckets with live replication enabled won't initiate replica events in the destination bucket. If you want to change the encryption type of objects in both your source and destination buckets, you must initiate separate `UpdateObjectEncryption` requests on the objects in the source and destination buckets.
- By default, all `UpdateObjectEncryption` requests that specify a customer-managed KMS key are restricted to KMS keys that are owned by the bucket owner's AWS account. If you're using AWS Organizations, you can request the ability to use AWS KMS keys owned by other member accounts within your organization by contacting AWS Support.
- If you use S3 Batch Replication to replicate datasets cross region and your objects previously had their server-side encryption type updated from SSE-S3 to SSE-KMS, you might need additional permissions. On the source region bucket, you must have `kms:decrypt` permissions. Then, you will need the `kms:decrypt` and `kms:encrypt` permissions for the bucket in the destination region.
- You must provide a full KMS key ARN in your `UpdateObjectEncryption` request. You can't use an alias name or alias ARN. You can determine the full KMS Key ARN in the AWS KMS Console or using the AWS KMS `DescribeKey` API.

Required permissions

To perform the `UpdateObjectEncryption` operation, you must have the following permissions:

- `s3:UpdateObjectEncryption`
- `kms:Encrypt`
- `kms:Decrypt`
- `kms:GenerateDataKey`
- `kms:ReEncrypt*`

If you're using AWS Organizations, to use this operation with customer-managed KMS keys from other AWS accounts within your organization, you must have the `organizations:DescribeAccount` permission. You must also request the ability to use AWS KMS keys owned by other member accounts within your organization by contacting AWS Support.

To perform the `UpdateObjectEncryption` operation, add the following AWS Identity and Access Management (IAM) policy to your IAM role. To use this policy, replace *amzn-s3-demo-bucket* with the name of your general purpose bucket, and replace the other *user input placeholders* with your own information.

```
{
    "Version": "2012-10-17",
    "Statement": [{
            "Sid": "AllowUpdateObjectEncryption",
            "Effect": "Allow",
            "Action": [
                "s3:UpdateObjectEncryption",
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:GenerateDataKey",
                "kms:ReEncrypt*",
                "organizations:DescribeAccount"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:kms:us-east-1:111122223333:key/01234567-89ab-
cdef-0123-456789abcdef"
            ]
        }
    ]
}
```

Updating encryption in bulk

To update the server-side encryption type of more than one Amazon S3 object with a single request, you can use S3 Batch Operations. You can provide S3 Batch Operations with a list of objects to operate on, or you can direct Batch Operations to generate an object list based object metadata, including prefix, storage class, creation date, encryption type, KMS key ARN, or S3 Bucket Key status. S3 Batch Operations calls the respective API operation to perform the specified operation. A single Batch Operations job can perform the specified operation on billions of objects within a bucket containing petabytes of data. For more information about Batch Operations, see the section called “Performing object operations in bulk”.

The S3 Batch Operations feature tracks progress, sends notifications, and stores a detailed completion report of all actions, providing a fully managed, auditable, serverless experience. You

can use S3 Batch Operations through the Amazon S3 console, AWS Command Line Interface (AWS CLI) AWS SDKs, or the Amazon S3 REST API. For more information, see the section called "Update object encryption".

Updating encryption for objects

You can update the server-side encryption type for an object through the AWS Command Line Interface (AWS CLI) AWS SDKs, or the Amazon S3 REST API.

Update encryption for an object

Using the AWS CLI

To run the following commands, you must have the AWS CLI installed and configured. If you don't have the AWS CLI installed, see Install or update to the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.

Alternatively, you can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? and Getting started with AWS CloudShell in the *AWS CloudShell User Guide*.

To update encryption for an object by using the AWS CLI

To use the following example command, replace the *`user input placeholders`* with your own information.

1. Use the following command to update encryption for a single object (*`index.html`*) in your general purpose bucket (for example, *`amzn-s3-demo-bucket`*) to use SSE-KMS with an S3 Bucket Key:

```
aws s3api update-object-encryption \
--bucket amzn-s3-demo-bucket \
--key index.html \
--object-encryption '{"SSEKMS": { "KMSKeyArn": "arn:aws:kms:us-
east-1:111122223333:key/f12a345a-678e-9bbb-1025-62e317037583", "BucketKeyEnabled":
 true }}'
```

> **Note**
>
> You must specify the full AWS KMS key Amazon Resource Name (ARN). The KMS key ID and KMS key alias aren't supported.

2. Run the `head-object` command to view the updated encryption type of your object:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

Using the REST API

You can send REST requests to update encryption for an object. For more information, see UpdateObjectEncryption.

Using the AWS SDKs

You can use the AWS SDKs to update encryption for an object. For more information, see the list of supported SDKs.

Java

Example

The following AWS SDK for Java 2.x example updates the encryption type to SSE-KMS for an object in a general purpose bucket.

```
    public void updateObjectEncryption(String bucketName,
                                       String objectKey,
                                       String versionId,
                                       String kmsKeyArn,
                                       boolean bucketKeyEnabled) {
        // Create the target object encryption type.
        ObjectEncryption objectEncryption = ObjectEncryption.builder()
                .ssekms(SSEKMSEncryption.builder()
                        .kmsKeyArn(kmsKeyArn)
                        .bucketKeyEnabled(bucketKeyEnabled)
                        .build())
                .build();

        // Create the UpdateObjectEncryption request.
```

```
        UpdateObjectEncryptionRequest request =
 UpdateObjectEncryptionRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
                .objectEncryption(objectEncryption)
                .build();

        // Update the object encryption.
        try {
            getS3Client().updateObjectEncryption(request);
            logger.info("Object encryption updated to SSE-KMS for {} in bucket {}",
 objectKey, bucketName);
        } catch (S3Exception e) {
            logger.error("Failed to update to object encryption: {} - Error code:
 {}", e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode());
            throw e;
        }
    }
```

Python

Example

The following AWS SDK for Python (Boto3) example shows how to update the encryption type to SSE-KMS for an object in a general purpose bucket.

```
response = client.update_object_encryption(
    Bucket='string',
    Key='string',
    VersionId='string',
    ObjectEncryption={
        'SSEKMS': {
                'KMSKeyArn': 'string',
                'BucketKeyEnabled': True|False
        }
    }
)
```

Using server-side encryption with Amazon S3 managed keys (SSE-S3)

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

All new object uploads to Amazon S3 buckets are encrypted by default with server-side encryption with Amazon S3 managed keys (SSE-S3).

Server-side encryption protects data at rest. Amazon S3 encrypts each object with a unique key. As an additional safeguard, it encrypts the key itself with a key that it rotates regularly. Amazon S3 server-side encryption uses 256-bit Advanced Encryption Standard Galois/Counter Mode (AES-GCM) to encrypt all uploaded objects.

There are no additional fees for using server-side encryption with Amazon S3 managed keys (SSE-S3). However, requests to configure the default encryption feature incur standard Amazon S3 request charges. For information about pricing, see Amazon S3 pricing.

If you require your data uploads to be encrypted using only Amazon S3 managed keys, you can use the following bucket policy. For example, the following bucket policy denies permissions to upload an object unless the request includes the `x-amz-server-side-encryption` header to request server-side encryption:

JSON

```
{
  "Version":"2012-10-17",
  "Id": "PutObjectPolicy",
  "Statement": [
    {
      "Sid": "DenyObjectsThatAreNotSSES3",
```

```
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
      "Condition": {
        "StringNotEquals": {
          "s3:x-amz-server-side-encryption": "AES256"
        }
      }
    }
  ]
}
```

Note

Server-side encryption encrypts only the object data, not the object metadata.

Note

The `s3:x-amz-server-side-encryption` condition key used in bucket policies is an IAM condition key namespace and differs from the `x-amz-server-side-encryption` HTTP header shown in CloudTrail logs. Both refer to the same encryption setting, but use different formats. Additionally, this Deny policy might block writes from (such as Elastic Load Balancing access logs or Amazon CloudFront logs) that deliver objects to your bucket without specifying the encryption header.

API support for server-side encryption

All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can update the default encryption configuration in the destination bucket.

If you want to specify a different encryption type in your PUT requests, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side

encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If you want to set a different default encryption configuration in the destination bucket, you can use SSE-KMS or DSSE-KMS.

For more information about changing the default encryption configuration for your general purpose buckets, see Configuring default encryption.

When you change the default encryption configuration of your bucket to SSE-KMS, the encryption type of the existing Amazon S3 objects in the bucket is not changed. To change the encryption type of your pre-existing objects after updating the default encryption configuration to SSE-KMS, you can use Amazon S3 Batch Operations. You provide S3 Batch Operations with a list of objects, and Batch Operations calls the respective API operation. You can use the Copy objects action to copy existing objects, which writes them back to the same bucket as SSE-KMS encrypted objects. A single Batch Operations job can perform the specified operation on billions of objects. For more information, see Performing object operations in bulk with Batch Operations and the *AWS Storage Blog* post How to retroactively encrypt existing objects in Amazon S3 using S3 Inventory, Amazon Athena, and S3 Batch Operations.

To configure server-side encryption by using the object creation REST APIs, you must provide the `x-amz-server-side-encryption` request header. For information about the REST APIs, see Using the REST API.

The following Amazon S3 APIs support this header:

- **PUT operations** – Specify the request header when uploading data using the PUT API. For more information, see PUT Object.
- **Initiate Multipart Upload** – Specify the header in the initiate request when uploading large objects using the multipart upload API operation. For more information, see Initiate Multipart Upload.
- **COPY operations** – When you copy an object, you have both a source object and a target object. For more information, see PUT Object - Copy.

> **Note**
>
> When using a POST operation to upload an object, instead of providing the request header, you provide the same information in the form fields. For more information, see POST Object.

The AWS SDKs also provide wrapper APIs that you can use to request server-side encryption. You can also use the AWS Management Console to upload objects and request server-side encryption.

For more general information, see AWS KMS concepts in the *AWS Key Management Service Developer Guide*.

Topics

- Specifying server-side encryption with Amazon S3 managed keys (SSE-S3)

Specifying server-side encryption with Amazon S3 managed keys (SSE-S3)

All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can update the default encryption configuration in the destination bucket.

If you want to specify a different encryption type in your PUT requests, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If you want to set a different default encryption configuration in the destination bucket, you can use SSE-KMS or DSSE-KMS.

For more information about changing the default encryption configuration for your general purpose buckets, see Configuring default encryption.

When you change the default encryption configuration of your bucket to SSE-KMS, the encryption type of the existing Amazon S3 objects in the bucket is not changed. To change the encryption type of your pre-existing objects after updating the default encryption configuration to SSE-KMS, you can use Amazon S3 Batch Operations. You provide S3 Batch Operations with a list of objects, and Batch Operations calls the respective API operation. You can use the Copy objects action to copy existing objects, which writes them back to the same bucket as SSE-KMS encrypted objects. A single Batch Operations job can perform the specified operation on billions of objects. For more information, see Performing object operations in bulk with Batch Operations and the *AWS Storage Blog* post How to retroactively encrypt existing objects in Amazon S3 using S3 Inventory, Amazon Athena, and S3 Batch Operations.

You can specify SSE-S3 by using the S3 console, REST APIs, AWS SDKs, and AWS Command Line Interface (AWS CLI). For more information, see Setting default server-side encryption behavior for Amazon S3 buckets.

Using the S3 console

This topic describes how to set or change the type of encryption an object by using the AWS Management Console. When you copy an object by using the console, Amazon S3 copies the object as is. That means that if the source object is encrypted, the target object is also encrypted. You can use the console to add or change encryption for an object.

Note

- You can change an object's encryption if your object is less than 5 GB. If your object is greater than 5 GB, you must use the AWS CLI or AWS SDKs to change an object's encryption.
- For a list of additional permissions required to change an object's encryption, see the section called "Required permissions for S3 API operations". For example policies that grant this permission, see the section called "Identity-based policy examples".
- If you change an object's encryption, a new object is created to replace the old one. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. The role that changes the property also becomes the owner of the new object (or object version).

To change encryption for an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Buckets**, and then choose the **General purpose buckets** tab. Navigate to the Amazon S3 bucket or folder that contains the objects you want to change.
3. Select the check box for the objects you want to change.
4. On the **Actions** menu, choose **Edit server-side encryption** from the list of options that appears.
5. Scroll to the **Server-side encryption** section.

6. Under **Encryption settings**, choose **Use bucket settings for default encryption** or **Override bucket settings for default encryption**.
7. If you chose **Override bucket settings for default encryption**, configure the following encryption settings.

    - Under **Encryption type**, choose **Server-side encryption with Amazon S3 managed keys (SSE-S3)**. SSE-S3 uses one of the strongest block ciphers—256-bit Advanced Encryption Standard (AES-256) to encrypt each object. For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).
8. Under **Additional copy settings**, choose whether you want to **Copy source settings**, **Don't specify settings**, or **Specify settings**. **Copy source settings** is the default option. If you only want to copy the object without the source settings attributes, choose **Don't specify settings**. Choose **Specify settings** to specify settings for storage class, ACLs, object tags, metadata, server-side encryption, and additional checksums.
9. Choose **Save changes**.

> **Note**
>
> This action applies encryption to all specified objects. When you're encrypting folders, wait for the save operation to finish before adding new objects to the folder.

Using the REST API

At the time of object creation—that is, when you are uploading a new object or making a copy of an existing object—you can specify if you want Amazon S3 to encrypt your data with Amazon S3 managed keys (SSE-S3) by adding the `x-amz-server-side-encryption` header to the request. Set the value of the header to the encryption algorithm `AES256`, which Amazon S3 supports. Amazon S3 confirms that your object is stored with SSE-S3 by returning the response header `x-amz-server-side-encryption`.

The following REST upload API operations accept the `x-amz-server-side-encryption` request header.

- PUT Object
- PUT Object - Copy
- POST Object

- Initiate Multipart Upload

When uploading large objects by using the multipart upload API operation, you can specify server-side encryption by adding the `x-amz-server-side-encryption` header to the Initiate Multipart Upload request. When you're copying an existing object, regardless of whether the source object is encrypted or not, the destination object is not encrypted unless you explicitly request server-side encryption.

The response headers of the following REST API operations return the `x-amz-server-side-encryption` header when an object is stored using SSE-S3.

- PUT Object
- PUT Object - Copy
- POST Object
- Initiate Multipart Upload
- Upload Part
- Upload Part - Copy
- Complete Multipart Upload
- Get Object
- Head Object

> **Note**
>
> Do not send encryption request headers for `GET` requests and `HEAD` requests if your object uses SSE-S3, or you'll get an HTTP status code 400 (Bad Request) error.

Using the AWS SDKs

When using AWS SDKs, you can request Amazon S3 to use server-side encryption with Amazon S3 managed encryption keys (SSE-S3). This section provides examples of using the AWS SDKs in multiple languages. For information about other SDKs, go to Sample Code and Libraries.

Java

When you use the AWS SDK for Java to upload an object, you can use SSE-S3 to encrypt it. To request server-side encryption, use the `ObjectMetadata` property of the `PutObjectRequest` to set the `x-amz-server-side-encryption` request header. When you call the `putObject()` method of the `AmazonS3Client`, Amazon S3 encrypts and saves the data.

You can also request SSE-S3 encryption when uploading objects with the multipart upload API operation:

- When using the high-level multipart upload API operation, you use the `TransferManager` methods to apply server-side encryption to objects as you upload them. You can use any of the upload methods that take `ObjectMetadata` as a parameter. For more information, see Uploading an object using multipart upload.
- When using the low-level multipart upload API operation, you specify server-side encryption when you initiate the multipart upload. You add the `ObjectMetadata` property by calling the `InitiateMultipartUploadRequest.setObjectMetadata()` method. For more information, see Using the AWS SDKs (low-level API).

You can't directly change the encryption state of an object (encrypting an unencrypted object or decrypting an encrypted object). To change an object's encryption state, you make a copy of the object, specifying the desired encryption state for the copy, and then delete the original object. Amazon S3 encrypts the copied object only if you explicitly request server-side encryption. To request encryption of the copied object through the Java API, use the `ObjectMetadata` property to specify server-side encryption in the `CopyObjectRequest`.

Example

The following example shows how to set server-side encryption by using the AWS SDK for Java. It shows how to perform the following tasks:

- Upload a new object by using SSE-S3.
- Change an object's encryption state (in this example, encrypting a previously unencrypted object) by making a copy of the object.
- Check the encryption state of the object.

For more information about server-side encryption, see Using the REST API. For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.internal.SSEResultBase;
import com.amazonaws.services.s3.model.*;

import java.io.ByteArrayInputStream;

public class SpecifyServerSideEncryption {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyNameToEncrypt = "*** Key name for an object to upload and encrypt
 ***";
        String keyNameToCopyAndEncrypt = "*** Key name for an unencrypted object to
 be encrypted by copying ***";
        String copiedObjectKeyName = "*** Key name for the encrypted copy of the
 unencrypted object ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();

            // Upload an object and encrypt it with SSE.
            uploadObjectWithSSEEncryption(s3Client, bucketName, keyNameToEncrypt);

            // Upload a new unencrypted object, then change its encryption state
            // to encrypted by making a copy.
            changeSSEEncryptionStatusByCopying(s3Client,
                    bucketName,
                    keyNameToCopyAndEncrypt,
                    copiedObjectKeyName);
```

```
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    private static void uploadObjectWithSSEEncryption(AmazonS3 s3Client, String
bucketName, String keyName) {
        String objectContent = "Test object encrypted with SSE";
        byte[] objectBytes = objectContent.getBytes();

        // Specify server-side encryption.
        ObjectMetadata objectMetadata = new ObjectMetadata();
        objectMetadata.setContentLength(objectBytes.length);

objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
        PutObjectRequest putRequest = new PutObjectRequest(bucketName,
                keyName,
                new ByteArrayInputStream(objectBytes),
                objectMetadata);

        // Upload the object and check its encryption status.
        PutObjectResult putResult = s3Client.putObject(putRequest);
        System.out.println("Object \"" + keyName + "\" uploaded with SSE.");
        printEncryptionStatus(putResult);
    }

    private static void changeSSEEncryptionStatusByCopying(AmazonS3 s3Client,
            String bucketName,
            String sourceKey,
            String destKey) {
        // Upload a new, unencrypted object.
        PutObjectResult putResult = s3Client.putObject(bucketName, sourceKey,
"Object example to encrypt by copying");
        System.out.println("Unencrypted object \"" + sourceKey + "\" uploaded.");
        printEncryptionStatus(putResult);

        // Make a copy of the object and use server-side encryption when storing the
        // copy.
```

```
        CopyObjectRequest request = new CopyObjectRequest(bucketName,
                sourceKey,
                bucketName,
                destKey);
        ObjectMetadata objectMetadata = new ObjectMetadata();

 objectMetadata.setSSEAlgorithm(ObjectMetadata.AES_256_SERVER_SIDE_ENCRYPTION);
        request.setNewObjectMetadata(objectMetadata);

        // Perform the copy operation and display the copy's encryption status.
        CopyObjectResult response = s3Client.copyObject(request);
        System.out.println("Object \"" + destKey + "\" uploaded with SSE.");
        printEncryptionStatus(response);

        // Delete the original, unencrypted object, leaving only the encrypted copy
 in
        // Amazon S3.
        s3Client.deleteObject(bucketName, sourceKey);
        System.out.println("Unencrypted object \"" + sourceKey + "\" deleted.");
    }

    private static void printEncryptionStatus(SSEResultBase response) {
        String encryptionStatus = response.getSSEAlgorithm();
        if (encryptionStatus == null) {
            encryptionStatus = "Not encrypted with SSE";
        }
        System.out.println("Object encryption status is: " + encryptionStatus);
    }
}
```

.NET

When you upload an object, you can direct Amazon S3 to encrypt it. To change the encryption state of an existing object, you make a copy of the object and delete the source object. By default, the copy operation encrypts the target only if you explicitly request server-side encryption of the target object. To specify SSE-S3 in the `CopyObjectRequest`, add the following:

```
ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
```

For a working sample of how to copy an object, see Using the AWS SDKs.

The following example uploads an object. In the request, the example directs Amazon S3 to encrypt the object. The example then retrieves object metadata and verifies the encryption method that was used. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SpecifyServerSideEncryptionTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for object created ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            WritingAnObjectAsync().Wait();
        }

        static async Task WritingAnObjectAsync()
        {
            try
            {
                var putRequest = new PutObjectRequest
                {
                    BucketName = bucketName,
                    Key = keyName,
                    ContentBody = "sample text",
                    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AES256
                };

                var putResponse = await client.PutObjectAsync(putRequest);

                // Determine the encryption state of an object.
```

```
                GetObjectMetadataRequest metadataRequest = new
 GetObjectMetadataRequest
                {
                    BucketName = bucketName,
                    Key = keyName
                };
                GetObjectMetadataResponse response = await
 client.GetObjectMetadataAsync(metadataRequest);
                ServerSideEncryptionMethod objectEncryption =
 response.ServerSideEncryptionMethod;

                Console.WriteLine("Encryption method used: {0}",
 objectEncryption.ToString());
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
 an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

This topic shows how to use classes from version 3 of the AWS SDK for PHP to add SSE-S3 to objects that you upload to Amazon S3. For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

To upload an object to Amazon S3, use the Aws\S3\S3Client::putObject() method. To add the `x-amz-server-side-encryption` request header to your upload request, specify the `ServerSideEncryption` parameter with the value `AES256`, as shown in the following code example. For information about server-side encryption requests, see Using the REST API.

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;
```

```
$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

// $filepath should be an absolute path to a file on disk.
$filepath = '*** Your File Path ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Upload a file with server-side encryption.
$result = $s3->putObject([
    'Bucket'               => $bucket,
    'Key'                  => $keyname,
    'SourceFile'           => $filepath,
    'ServerSideEncryption' => 'AES256',
]);
```

In response, Amazon S3 returns the `x-amz-server-side-encryption` header with the value of the encryption algorithm that was used to encrypt your object's data.

When you upload large objects by using the multipart upload API operation, you can specify SSE-S3 for the objects that you are uploading, as follows:

- When you're using the low-level multipart upload API operation, specify server-side encryption when you call the Aws\S3\S3Client::createMultipartUpload() method. To add the `x-amz-server-side-encryption` request header to your request, specify the `array` parameter's `ServerSideEncryption` key with the value AES256. For more information about the low-level multipart upload API operation, see Using the AWS SDKs (low-level API).
- When you're using the high-level multipart upload API operation, specify server-side encryption by using the `ServerSideEncryption` parameter of the CreateMultipartUpload API operation. For an example of using the `setOption()` method with the high-level multipart upload API operation, see Uploading an object using multipart upload.

To determine the encryption state of an existing object, retrieve the object metadata by calling the Aws\S3\S3Client::headObject() method as shown in the following PHP code example.

```
require 'vendor/autoload.php';
```

```
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';
$keyname = '*** Your Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Check which server-side encryption algorithm is used.
$result = $s3->headObject([
    'Bucket' => $bucket,
    'Key'    => $keyname,
]);
echo $result['ServerSideEncryption'];
```

To change the encryption state of an existing object, make a copy of the object by using the Aws\S3\S3Client::copyObject() method and delete the source object. By default, `copyObject()` does not encrypt the target unless you explicitly request server-side encryption of the destination object by using the `ServerSideEncryption` parameter with the value AES256. The following PHP code example makes a copy of an object and adds server-side encryption to the copied object.

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$sourceBucket = '*** Your Source Bucket Name ***';
$sourceKeyname = '*** Your Source Object Key ***';

$targetBucket = '*** Your Target Bucket Name ***';
$targetKeyname = '*** Your Target Object Key ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);

// Copy an object and add server-side encryption.
```

```
$s3->copyObject([
    'Bucket'               => $targetBucket,
    'Key'                  => $targetKeyname,
    'CopySource'           => "$sourceBucket/$sourceKeyname",
    'ServerSideEncryption' => 'AES256',
]);
```

For more information, see the following topics:

- AWS SDK for PHP for Amazon S3 Aws\S3\S3Client Class
- AWS SDK for PHP Documentation

Ruby

When using the AWS SDK for Ruby to upload an object, you can specify that the object be stored encrypted at rest with SSE-S3. When you read the object back, it is automatically decrypted.

The following AWS SDK for Ruby Version 3 example demonstrates how to specify that a file uploaded to Amazon S3 be encrypted at rest.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
 object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
 #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

The following code example demonstrates how to determine the encryption state of an existing object.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
```

```
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
 object_key))
  obj_data = wrapper.get_object
  return unless obj_data

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
 obj_data.server_side_encryption
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

If server-side encryption is not used for the object that is stored in Amazon S3, the method returns `null`.

To change the encryption state of an existing object, make a copy of the object and delete the source object. By default, the copy methods do not encrypt the target unless you explicitly request server-side encryption. You can request the encryption of the target object by specifying the `server_side_encryption` value in the option's hash argument, as shown in the following Ruby code example. The code example demonstrates how to copy an object and encrypt the copy with SSE-S3.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                       copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
```

```
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key, target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
 #{target_object.bucket_name}:#{target_object.key} and "\
       "encrypted the target with #{target_object.server_side_encryption}
 encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Using the AWS CLI

To specify SSE-S3 when you upload an object by using the AWS CLI, use the following example.

```
aws s3api put-object --bucket amzn-s3-demo-bucket1 --key object-key-name --server-side-
encryption AES256  --body file path
```

For more information, see put-object in the *AWS CLI reference*. To specify SSE-S3 when you copy an object by using the AWS CLI, see copy-object.

Using CloudFormation

For examples of setting up encryption using CloudFormation, see Create a bucket with default encryption and the Create a bucket by using AWS KMS server-side encryption with an S3 Bucket Key example in the `AWS::S3::Bucket ServerSideEncryptionRule` topic in the *AWS CloudFormation User Guide*.

Using server-side encryption with AWS KMS keys (SSE-KMS)

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

Server-side encryption is the encryption of data at its destination by the application or service that receives it.

Amazon S3 automatically enables server-side encryption with Amazon S3 managed keys (SSE-S3) for new object uploads.

Unless you specify otherwise, buckets use SSE-S3 by default to encrypt objects. However, you can choose to configure buckets to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) instead. For more information, see Specifying server-side encryption with AWS KMS (SSE-KMS).

AWS KMS is a service that combines secure, highly available hardware and software to provide a key management system scaled for the cloud. Amazon S3 uses server-side encryption with

AWS KMS (SSE-KMS) to encrypt your S3 object data. Also, when SSE-KMS is requested for the object, the S3 checksum (as part of the object's metadata) is stored in encrypted form. For more information about checksum, see Checking object integrity in Amazon S3.

If you use KMS keys, you can use AWS KMS through the AWS Management Console or the AWS KMS API to do the following:

- Centrally create, view, edit, monitor, enable or disable, rotate, and schedule deletion of KMS keys.
- Define the policies that control how and by whom KMS keys can be used.
- Audit KMS key usage for correct use. Auditing is supported by the AWS KMS API but not by the AWS KMS Console;.

The security controls in AWS KMS can help you meet encryption-related compliance requirements. You can use these KMS keys to protect your data in Amazon S3 buckets. When you use SSE-KMS encryption with an S3 bucket, the AWS KMS keys must be in the same Region as the bucket.

There are additional charges for using AWS KMS keys. For more information, see AWS KMS key concepts in the *AWS Key Management Service Developer Guide* and AWS KMS pricing.

For instructions on allowing IAM users to access KMS-encrypted buckets see My Amazon S3 bucket has default encryption using a custom AWS KMS key. How can I allow users to download from and upload to the bucket? in the AWS re:Post Knowledge Center.

Permissions

To successfully make a `PutObject` request to encrypt an object with an AWS KMS key to Amazon S3, you need `kms:GenerateDataKey` permissions on the key. To download an object encrypted with an AWS KMS key, you need `kms:Decrypt` permissions for the key. To perform a multipart upload to encrypt an object with an AWS KMS key, you must have the `kms:GenerateDataKey` and `kms:Decrypt` permissions for the key.

Important

- Carefully review the permissions that are granted in your KMS key policies. Always restrict customer-managed KMS key policy permissions only to the IAM principals and AWS services that must access the relevant AWS KMS key action. For more information, see Key policies in AWS KMS.

- If a bucket is used as a destination for Amazon S3 server access logging, the destination bucket must use Amazon S3 managed keys (SSE-S3). If the destination bucket uses SSE-KMS default encryption, Amazon S3 might deliver log objects that are encrypted with a key that you can't access. To resolve this, change the destination bucket's default encryption to SSE-S3. For more information, see Enabling Amazon S3 server access logging.

Topics

- AWS KMS keys
- Amazon S3 Bucket Keys
- Requiring server-side encryption
- Encryption context
- Sending requests for AWS KMS encrypted objects
- Specifying server-side encryption with AWS KMS (SSE-KMS)
- Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys

AWS KMS keys

When you use server-side encryption with AWS KMS (SSE-KMS), you can use the default AWS managed key, or you can specify a customer managed key that you have already created. AWS KMS supports *envelope encryption*. S3 uses the AWS KMS features for *envelope encryption* to further protect your data. Envelope encryption is the practice of encrypting your plain text data with a data key, and then encrypting that data key with a KMS key. For more information about envelope encryption, see Envelope encryption in the *AWS Key Management Service Developer Guide*.

If you don't specify a customer managed key, Amazon S3 automatically creates an AWS managed key in your AWS account the first time that you add an object encrypted with SSE-KMS to a bucket. By default, Amazon S3 uses this KMS key for SSE-KMS.

Note

Objects encrypted using SSE-KMS with AWS managed keys can't be shared cross-account. If you need to share SSE-KMS data cross-account, you must use a customer managed key from AWS KMS.

If you want to use a customer managed key for SSE-KMS, create a symmetric encryption customer managed key before you configure SSE-KMS. Then, when you configure SSE-KMS for your bucket, specify the existing customer managed key. For more information about symmetric encryption key, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

Creating a customer managed key gives you more flexibility and control. For example, you can create, rotate, and disable customer managed keys. You can also define access controls and audit the customer managed key that you use to protect your data. For more information about customer managed and AWS managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.

> **Note**
>
> When you use server-side encryption with a customer managed key that's stored in an external key store, unlike standard KMS keys, you are responsible for ensuring the availability and durability of your key material. For more information about external key stores and how they shift the shared responsibility model, see External key stores in the *AWS Key Management Service Developer Guide*.

Using SSE-KMS encryption for cross-account operations

When using encryption for cross-account operations, be aware of the following:

- If an AWS KMS key Amazon Resource Name (ARN) or alias is not provided at request time or through the bucket's default encryption configuration, the AWS managed key (`aws/s3`) from the uploading account is used for encryption and required for decryption.
- AWS managed key (`aws/s3`) can be used as your KMS key for cross-account operations when the uploading and accessing AWS Identity and Access Management (IAM) principals are from the same AWS account.
- If you want to grant cross-account access to your S3 objects, use a customer managed key. You can configure the policy of a customer managed key to allow access from another account.
- If you're specifying a customer managed KMS key, we recommend using a fully qualified KMS key ARN. If you use a KMS key alias instead, AWS KMS resolves the key within the requester's account. This behavior can result in data that's encrypted with a KMS key that belongs to the requester, and not the bucket owner.

- You must specify a key that you (the requester) have been granted `Encrypt` permission to. For more information, see Allow key users to use a KMS key for cryptographic operations in the *AWS Key Management Service Developer Guide*.

For more information about when to use customer managed keys and AWS managed KMS keys, see Should I use an AWS managed key or a customer managed key to encrypt my objects in Amazon S3?

SSE-KMS encryption workflow

If you choose to encrypt your data using an AWS managed key or a customer managed key, AWS KMS and Amazon S3 perform the following envelope encryption actions:

1. Amazon S3 requests a plaintext data key and a copy of the key encrypted under the specified KMS key.
2. AWS KMS generates a data key, encrypts it under the KMS key, and sends both the plaintext data key and the encrypted data key to Amazon S3.
3. Amazon S3 encrypts the data using the data key and removes the plaintext key from memory as soon as possible after use.
4. Amazon S3 stores the encrypted data key as metadata with the encrypted data.

When you request that your data be decrypted, Amazon S3 and AWS KMS perform the following actions:

1. Amazon S3 sends the encrypted data key to AWS KMS in a `Decrypt` request.
2. AWS KMS decrypts the encrypted data key by using the same KMS key and returns the plaintext data key to Amazon S3.
3. Amazon S3 decrypts the encrypted data, using the plaintext data key, and removes the plaintext data key from memory as soon as possible.

Important

When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS

keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

Auditing SSE-KMS encryption

To identify requests that specify SSE-KMS, you can use the **All SSE-KMS requests** and **% all SSE-KMS requests** metrics in Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. You can also use the SSE-KMS enabled bucket count and % SSE-KMS enabled buckets to understand the count of buckets that (SSE-KMS) for default bucket encryption. For more information, see Assessing your storage activity and usage with S3 Storage Lens. For a complete list of metrics, see S3 Storage Lens metrics glossary.

To audit the usage of your AWS KMS keys for your SSE-KMS encrypted data, you can use AWS CloudTrail logs. You can get insight into your cryptographic operations, such as GenerateDataKey and Decrypt. CloudTrail supports numerous attribute values for filtering your search, including event name, user name, and event source.

Amazon S3 Bucket Keys

When you configure server-side encryption using AWS KMS (SSE-KMS), you can configure your buckets to use S3 Bucket Keys for SSE-KMS. Using a bucket-level key for SSE-KMS can reduce your AWS KMS request costs by up to 99 percent by decreasing the request traffic from Amazon S3 to AWS KMS.

When you configure a bucket to use an S3 Bucket Key for SSE-KMS on new objects, AWS KMS generates a bucket-level key that is used to create unique data keys for objects in the bucket. This S3 Bucket Key is used for a time-limited period within Amazon S3, further reducing the need for Amazon S3 to make requests to AWS KMS to complete encryption operations. For more information about using S3 Bucket Keys, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

Requiring server-side encryption

To require server-side encryption of all objects in a particular Amazon S3 bucket, you can use a bucket policy. For example, the following bucket policy denies the upload object (`s3:PutObject`) permission to everyone if the request does not include an `x-amz-server-side-encryption-aws-kms-key-id` header that requests server-side encryption with SSE-KMS.

JSON

```
{
   "Version":"2012-10-17",
   "Id":"PutObjectPolicy",
   "Statement":[{
         "Sid":"DenyObjectsThatAreNotSSEKMS",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3:PutObject",
         "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*",
         "Condition":{
            "Null":{
               "s3:x-amz-server-side-encryption-aws-kms-key-id":"true"
            }
         }
      }
   ]
}
```

To require that a particular AWS KMS key be used to encrypt the objects in a bucket, you can use the `s3:x-amz-server-side-encryption-aws-kms-key-id` condition key. To specify the KMS key, you must use a key Amazon Resource Name (ARN) that is in the `arn:aws:kms:`*`region`*`:`*`acct-id`*`:key/`*`key-id`* format. AWS Identity and Access Management does not validate if the string for `s3:x-amz-server-side-encryption-aws-kms-key-id` exists.

Note

When you upload an object, you can specify the KMS key by using the `x-amz-server-side-encryption-aws-kms-key-id` header or rely on your default bucket encryption configuration. If your PutObject request specifies `aws:kms` in the `x-amz-server-side-encryption` header, but does not specify the `x-amz-server-side-encryption-aws-kms-key-id` header, then Amazon S3 assumes that you want to use the AWS managed key. Regardless, the AWS KMS key ID that Amazon S3 uses for object encryption must match the AWS KMS key ID in the policy, otherwise Amazon S3 denies the request.

For a complete list of Amazon S3 specific condition keys, see Condition keys for Amazon S3 in the *Service Authorization Reference*.

Encryption context

An *encryption context* is a set of key-value pairs that contains additional contextual information about the data. The encryption context is not encrypted. When an encryption context is specified for an encryption operation, Amazon S3 must specify the same encryption context for the decryption operation. Otherwise, the decryption fails. AWS KMS uses the encryption context as additional authenticated data (AAD) to support authenticated encryption. For more information about the encryption context, see Encryption context in the *AWS Key Management Service Developer Guide*.

By default, Amazon S3 uses the object or bucket Amazon Resource Name (ARN) as the encryption context pair:

- **If you use SSE-KMS without enabling an S3 Bucket Key**, the object ARN is used as the encryption context.

```
arn:aws:s3:::object_ARN
```

- **If you use SSE-KMS and enable an S3 Bucket Key**, the bucket ARN is used as the encryption context. For more information about S3 Bucket Keys, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

```
arn:aws:s3:::bucket_ARN
```

You can optionally provide an additional encryption context pair by using the `x-amz-server-side-encryption-context` header in an s3:PutObject request. However, because the encryption context is not encrypted, make sure it does not include sensitive information. Amazon S3 stores this additional key pair alongside the default encryption context. When it processes your PUT request, Amazon S3 appends the default encryption context of `aws:s3:arn` to the one that you provide.

You can use the encryption context to identify and categorize your cryptographic operations. You can also use the default encryption context ARN value to track relevant requests in AWS CloudTrail by viewing which Amazon S3 ARN was used with which encryption key.

In the `requestParameters` field of a CloudTrail log file, the encryption context looks similar to the following one.

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket1/file_name"
}
```

When you use SSE-KMS with the optional S3 Bucket Keys feature, the encryption context value is the ARN of the bucket.

```
"encryptionContext": {
    "aws:s3:arn": "arn:aws:s3:::amzn-s3-demo-bucket1"
}
```

Sending requests for AWS KMS encrypted objects

> **Important**
>
> All GET and PUT requests for AWS KMS encrypted objects must be made using Secure Sockets Layer (SSL) or Transport Layer Security (TLS). Requests must also be signed using valid credentials, such as AWS Signature Version 4 (or AWS Signature Version 2).

AWS Signature Version 4 is the process of adding authentication information to AWS requests sent by HTTP. For security, most requests to AWS must be signed with an access key, which consists of an access key ID and secret access key. These two keys are commonly referred to as your security credentials. For more information, see Authenticating Requests (AWS Signature Version 4) and Signature Version 4 signing process.

> **Important**
>
> If your object uses SSE-KMS, don't send encryption request headers for GET requests and HEAD requests. Otherwise, you'll get an HTTP 400 Bad Request error.

Topics

- Specifying server-side encryption with AWS KMS (SSE-KMS)
- Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys

Specifying server-side encryption with AWS KMS (SSE-KMS)

All Amazon S3 buckets have encryption configured by default, and all new objects that are uploaded to an S3 bucket are automatically encrypted at rest. Server-side encryption with Amazon S3 managed keys (SSE-S3) is the default encryption configuration for every bucket in Amazon S3. To use a different type of encryption, you can either specify the type of server-side encryption to use in your S3 PUT requests, or you can update the default encryption configuration in the destination bucket.

If you want to specify a different encryption type in your PUT requests, you can use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). If you want to set a different default encryption configuration in the destination bucket, you can use SSE-KMS or DSSE-KMS.

For more information about changing the default encryption configuration for your general purpose buckets, see Configuring default encryption.

When you change the default encryption configuration of your bucket to SSE-KMS, the encryption type of the existing Amazon S3 objects in the bucket is not changed. To change the encryption type of your pre-existing objects after updating the default encryption configuration to SSE-KMS, you can use Amazon S3 Batch Operations. You provide S3 Batch Operations with a list of objects, and Batch Operations calls the respective API operation. You can use the Copy objects action to copy existing objects, which writes them back to the same bucket as SSE-KMS encrypted objects. A single Batch Operations job can perform the specified operation on billions of objects. For more information, see Performing object operations in bulk with Batch Operations and the *AWS Storage Blog* post How to retroactively encrypt existing objects in Amazon S3 using S3 Inventory, Amazon Athena, and S3 Batch Operations.

You can specify SSE-KMS by using the Amazon S3 console, REST API operations, AWS SDKs, and the AWS Command Line Interface (AWS CLI). For more information, see the following topics.

> **Note**
>
> You can use multi-Region AWS KMS keys in Amazon S3. However, Amazon S3 currently treats multi-Region keys as though they were single-Region keys, and does not use the multi-Region features of the key. For more information, see Using multi-Region keys in the *AWS Key Management Service Developer Guide*.

Note

If you want to use a KMS key that's owned by a different account, you must have permission to use the key. For more information about cross-account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*.

Using the S3 console

This topic describes how to set or change the type of encryption of an object to use server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) by using the Amazon S3 console.

Note

- You can change an object's encryption if your object is less than 5 GB. If your object is greater than 5 GB, you must use the AWS CLI or AWS SDKs to change an object's encryption.
- For a list of additional permissions required to change an object's encryption, see the section called "Required permissions for S3 API operations". For example policies that grant this permission, see the section called "Identity-based policy examples".
- If you change an object's encryption, a new object is created to replace the old one. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. The role that changes the property also becomes the owner of the new object (or object version).

To add or change encryption for an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Buckets**, and then choose the **General purpose buckets** tab. Navigate to the Amazon S3 bucket or folder that contains the objects you want to change.
3. Select the check box for the objects you want to change.
4. On the **Actions** menu, choose **Edit server-side encryption** from the list of options that appears.

5. Scroll to the **Server-side encryption** section.
6. Under **Encryption settings**, choose **Use bucket settings for default encryption** or **Override bucket settings for default encryption**.

> **Important**
>
> If you use the SSE-KMS option for your default encryption configuration, you are subject to the requests per second (RPS) quotas of AWS KMS. For more information about AWS KMS quotas and how to request a quota increase, see Quotas in the *AWS Key Management Service Developer Guide*.

7. If you chose **Override bucket settings for default encryption**, configure the following encryption settings.

   a. Under **Encryption type**, choose **Server-side encryption with AWS Key Management Service keys (SSE-KMS)**.

   b. Under **AWS KMS key**, do one of the following to choose your KMS key:

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose your **KMS key** from the list of available keys.

     Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
   - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and then enter your KMS key ARN in the field that appears.
   - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

     For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that is not listed, you must enter your KMS

key ARN. If you want to use a KMS key that is owned by a different account, you must first have permission to use the key and then you must enter the KMS key ARN.
Amazon S3 supports only symmetric encryption KMS keys, and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

8. Under **Additional copy settings**, choose whether you want to **Copy source settings**, **Don't specify settings**, or **Specify settings**. **Copy source settings** is the default option. If you only want to copy the object without the source settings attributes, choose **Don't specify settings**. Choose **Specify settings** to specify settings for storage class, ACLs, object tags, metadata, server-side encryption, and additional checksums.
9. Choose **Save changes**.

Note

This action applies encryption to all specified objects. When you're encrypting folders, wait for the save operation to finish before adding new objects to the folder.

Using the REST API

When you create an object—that is, when you upload a new object or copy an existing object—you can specify the use of server-side encryption with AWS KMS keys (SSE-KMS) to encrypt your data. To do this, add the `x-amz-server-side-encryption` header to the request. Set the value of the header to the encryption algorithm `aws:kms`. Amazon S3 confirms that your object is stored using SSE-KMS by returning the response header `x-amz-server-side-encryption`.

If you specify the `x-amz-server-side-encryption` header with a value of `aws:kms`, you can also use the following request headers:

- `x-amz-server-side-encryption-aws-kms-key-id`
- `x-amz-server-side-encryption-context`
- `x-amz-server-side-encryption-bucket-key-enabled`

Topics

- Amazon S3 REST API operations that support SSE-KMS
- Encryption context (x-amz-server-side-encryption-context)
- AWS KMS key ID (x-amz-server-side-encryption-aws-kms-key-id)
- S3 Bucket Keys (x-amz-server-side-encryption-aws-bucket-key-enabled)

Amazon S3 REST API operations that support SSE-KMS

The following REST API operations accept the `x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id`, and `x-amz-server-side-encryption-context` request headers.

- PutObject – When you upload data by using the PUT API operation, you can specify these request headers.
- CopyObject – When you copy an object, you have both a source object and a target object. When you pass SSE-KMS headers with the `CopyObject` operation, they're applied only to the target object. When you're copying an existing object, regardless of whether the source object is encrypted or not, the destination object isn't encrypted unless you explicitly request server-side encryption.
- POST Object – When you use a POST operation to upload an object, instead of the request headers, you provide the same information in the form fields.
- CreateMultipartUpload – When you upload large objects by using the multipart upload API operation, you can specify these headers. You specify these headers in the `CreateMultipartUpload` request.

The response headers of the following REST API operations return the `x-amz-server-side-encryption` header when an object is stored by using server-side encryption.

- PutObject
- CopyObject
- POST Object
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload

- GetObject
- HeadObject

> **Important**
>
> - All GET and PUT requests for an object protected by AWS KMS fail if you don't make these requests by using Secure Sockets Layer (SSL), Transport Layer Security (TLS), or Signature Version 4.
> - If your object uses SSE-KMS, don't send encryption request headers for GET requests and HEAD requests, or you'll get an HTTP 400 BadRequest error.

Encryption context (`x-amz-server-side-encryption-context`)

If you specify `x-amz-server-side-encryption:aws:kms`, the Amazon S3 API supports an encryption context with the `x-amz-server-side-encryption-context` header. An encryption context is a set of key-value pairs that contain additional contextual information about the data.

Amazon S3 automatically uses the object or bucket Amazon Resource Name (ARN) as the encryption context pair. If you use SSE-KMS without enabling an S3 Bucket Key, you use the object ARN as your encryption context; for example, `arn:aws:s3:::`*`object_ARN`*. However, if you use SSE-KMS and enable an S3 Bucket Key, you use the bucket ARN for your encryption context; for example, `arn:aws:s3:::`*`bucket_ARN`*.

You can optionally provide an additional encryption context pair by using the `x-amz-server-side-encryption-context` header. However, because the encryption context isn't encrypted, make sure it doesn't include sensitive information. Amazon S3 stores this additional key pair alongside the default encryption context.

For information about the encryption context in Amazon S3, see Encryption context. For general information about the encryption context, see AWS Key Management Service Concepts - Encryption context in the *AWS Key Management Service Developer Guide*.

AWS KMS key ID (`x-amz-server-side-encryption-aws-kms-key-id`)

You can use the `x-amz-server-side-encryption-aws-kms-key-id` header to specify the ID of the customer managed key that's used to protect the data. If you specify the `x-amz-server-`

`side-encryption:aws:kms` header but don't provide the `x-amz-server-side-encryption-aws-kms-key-id` header, Amazon S3 uses the AWS managed key (`aws/s3`) to protect the data. If you want to use a customer managed key, you must provide the `x-amz-server-side-encryption-aws-kms-key-id` header of the customer managed key.

> **Important**
>
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

S3 Bucket Keys (`x-amz-server-side-encryption-aws-bucket-key-enabled`)

You can use the `x-amz-server-side-encryption-aws-bucket-key-enabled` request header to enable or disable an S3 Bucket Key at the object level. S3 Bucket Keys reduce your AWS KMS request costs by decreasing the request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

If you specify the `x-amz-server-side-encryption:aws:kms` header but don't provide the `x-amz-server-side-encryption-aws-bucket-key-enabled` header, your object uses the S3 Bucket Key settings for the destination bucket to encrypt your object. For more information, see Configuring an S3 Bucket Key at the object level.

Using the AWS CLI

To use the following example AWS CLI commands, replace the *`user input placeholders`* with your own information.

When you upload a new object or copy an existing object, you can specify the use of server-side encryption with AWS KMS keys to encrypt your data. To do this, add the `--server-side-encryption aws:kms` header to the request. Use the `--ssekms-key-id` *`example-key-id`* to add your customer managed AWS KMS key that you created. If you specify `--server-side-encryption aws:kms`, but don't provide an AWS KMS key ID, Amazon S3 will use an AWS managed key.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key example-object-key --server-side-encryption aws:kms --ssekms-key-id example-key-id --body filepath
```

You can additionally enable or disable Amazon S3 Bucket Keys on your PUT or COPY operations by adding `--bucket-key-enabled` or `--no-bucket-key-enabled`. Amazon S3 Bucket Keys can reduce your AWS KMS request costs by decreasing the request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key example-object-key --server-
side-encryption aws:kms --bucket-key-enabled --body filepath
```

You can encrypt an unencrypted object to use SSE-KMS by copying the object back in place.

```
aws s3api copy-object --bucket amzn-s3-demo-bucket --key example-object-key --
body filepath --bucket amzn-s3-demo-bucket --key example-object-key --sse aws:kms --
sse-kms-key-id example-key-id --body filepath
```

Using the AWS SDKs

When using AWS SDKs, you can request Amazon S3 to use AWS KMS keys for server-side encryption. The following examples show how to use SSE-KMS with the AWS SDKs for Java and .NET. For information about other SDKs, see Sample code and libraries on the AWS Developer Center.

> **Important**
>
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

`CopyObject` operation

When copying objects, you add the same request properties (`ServerSideEncryptionMethod` and `ServerSideEncryptionKeyManagementServiceKeyId`) to request Amazon S3 to use an AWS KMS key. For more information about copying objects, see Copying, moving, and renaming objects.

PUT operation

Java

When uploading an object by using the AWS SDK for Java, you can request Amazon S3 to use an AWS KMS key by adding the `SSEAwsKeyManagementParams` property as shown in the following request:

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new SSEAwsKeyManagementParams());
```

In this case, Amazon S3 uses the AWS managed key (`aws/s3`). For more information, see Using server-side encryption with AWS KMS keys (SSE-KMS). You can optionally create a symmetric encryption KMS key and specify that in the request, as shown in the following example:

```
PutObjectRequest putRequest = new PutObjectRequest(bucketName,
   keyName, file).withSSEAwsKeyManagementParams(new
 SSEAwsKeyManagementParams(keyID));
```

For more information about creating customer managed keys, see Programming the AWS KMS API in the *AWS Key Management Service Developer Guide*.

For working code examples of uploading an object, see the following topics. To use these examples, you must update the code examples and provide encryption information as shown in the preceding code fragment.

- For uploading an object in a single operation, see Uploading objects.
- For multipart uploads that use the high-level or low-level multipart upload API operations, see Uploading an object using multipart upload.

.NET

When uploading an object by using the AWS SDK for .NET, you can request Amazon S3 to use an AWS KMS key by adding the `ServerSideEncryptionMethod` property as shown in the following request:

```
PutObjectRequest putRequest = new PutObjectRequest
 {
```

```
    BucketName = amzn-s3-demo-bucket,
    Key = keyName,
    // other properties
    ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS
};
```

In this case, Amazon S3 uses the AWS managed key. For more information, see Using server-side encryption with AWS KMS keys (SSE-KMS). You can optionally create your own symmetric encryption customer managed key and specify that in the request, as shown in the following example:

```
PutObjectRequest putRequest1 = new PutObjectRequest
{
  BucketName = amzn-s3-demo-bucket,
  Key = keyName,
  // other properties
  ServerSideEncryptionMethod = ServerSideEncryptionMethod.AWSKMS,
  ServerSideEncryptionKeyManagementServiceKeyId = keyId
};
```

For more information about creating customer managed keys, see Programming the AWS KMS API in the *AWS Key Management Service Developer Guide*.

For working code examples of uploading an object, see the following topics. To use these examples, you must update the code examples and provide encryption information as shown in the preceding code fragment.

- For uploading an object in a single operation, see Uploading objects.
- For multipart uploads that use the high-level or low-level multipart upload API operations, see Uploading an object using multipart upload.

Presigned URLs

Java

When creating a presigned URL for an object that's encrypted with an AWS KMS key, you must explicitly specify Signature Version 4, as shown in the following example:

```
ClientConfiguration clientConfiguration = new ClientConfiguration();
```

```
clientConfiguration.setSignerOverride("AWSS3V4SignerType");
AmazonS3Client s3client = new AmazonS3Client(
        new ProfileCredentialsProvider(), clientConfiguration);
...
```

For a code example, see Sharing objects with presigned URLs.

.NET

When creating a presigned URL for an object that's encrypted with an AWS KMS key, you must explicitly specify Signature Version 4, as shown in the following example:

```
AWSConfigs.S3Config.UseSignatureVersion4 = true;
```

For a code example, see Sharing objects with presigned URLs.

Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys

Amazon S3 Bucket Keys reduce the cost of Amazon S3 server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS). Using a bucket-level key for SSE-KMS can reduce AWS KMS request costs by up to 99 percent by decreasing the request traffic from Amazon S3 to AWS KMS. With a few clicks in the AWS Management Console, and without any changes to your client applications, you can configure your bucket to use an S3 Bucket Key for SSE-KMS encryption on new objects.

> **Note**
>
> S3 Bucket Keys aren't supported for dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS).

S3 Bucket Keys for SSE-KMS

Workloads that access millions or billions of objects encrypted with SSE-KMS can generate large volumes of requests to AWS KMS. When you use SSE-KMS to protect your data without an S3 Bucket Key, Amazon S3 uses an individual AWS KMS data key for every object. In this case, Amazon S3 makes a call to AWS KMS every time a request is made against a KMS-encrypted object. For information about how SSE-KMS works, see Using server-side encryption with AWS KMS keys (SSE-KMS).

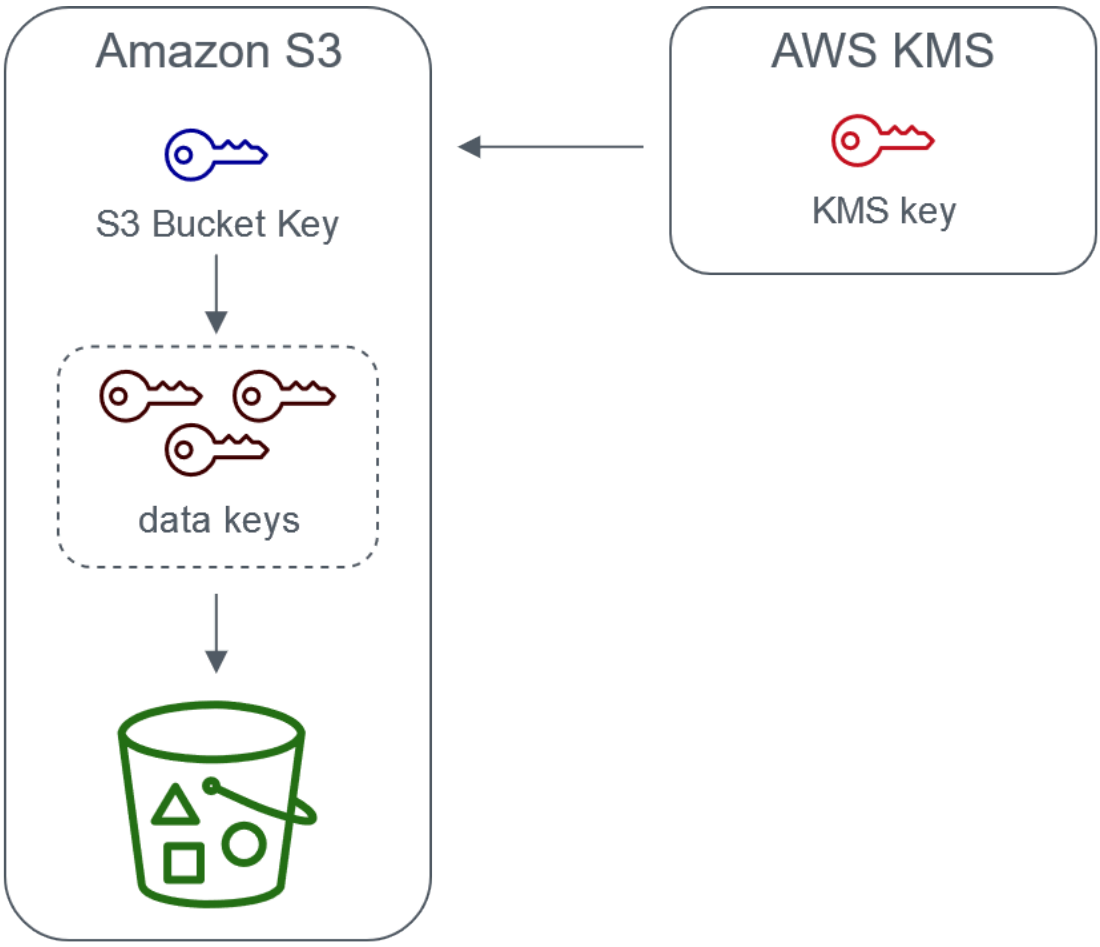
When you configure your bucket to use an S3 Bucket Key for SSE-KMS, AWS generates a short-lived bucket-level key from AWS KMS, then temporarily keeps it in S3. This bucket-level key will create data keys for new objects during its lifecycle. S3 Bucket Keys are used for a limited time period within Amazon S3, reducing the need for S3 to make requests to AWS KMS to complete encryption operations. This reduces traffic from S3 to AWS KMS, allowing you to access AWS KMS-encrypted objects in Amazon S3 at a fraction of the previous cost.

Unique bucket-level keys are fetched at least once per requester to ensure that the requester's access to the key is captured in an AWS KMS CloudTrail event. Amazon S3 treats callers as different requesters when they use different roles or accounts, or the same role with different scoping policies. AWS KMS request savings reflect the number of requesters, request patterns, and relative age of the objects requested. For example, a fewer number of requesters, requesting multiple objects in a limited time window, and encrypted with the same bucket-level key, results in greater savings.

> **Note**
>
> Using S3 Bucket Keys allows you to save on AWS KMS request costs by decreasing your requests to AWS KMS for `Encrypt`, `GenerateDataKey`, and `Decrypt` operations through the use of a bucket-level key. By design, subsequent requests that take advantage of this bucket-level key do not result in AWS KMS API requests or validate access against the AWS KMS key policy.

When you configure an S3 Bucket Key, objects that are already in the bucket do not use the S3 Bucket Key. To configure an S3 Bucket Key for existing objects, you can use a `CopyObject` operation. For more information, see Configuring an S3 Bucket Key at the object level.

Amazon S3 will only share an S3 Bucket Key for objects encrypted by the same AWS KMS key. S3 Bucket Keys are compatible with KMS keys created by AWS KMS, imported key material, and key material backed by custom key stores.



Server-side encryption with AWS Key Management service using an S3 Bucket Key

Configuring S3 Bucket Keys

You can configure your bucket to use an S3 Bucket Key for SSE-KMS on new objects through the Amazon S3 console, AWS SDKs, AWS CLI, or REST API. With S3 Bucket Keys enabled on your bucket, objects uploaded with a different specified SSE-KMS key will use their own S3 Bucket Keys. Regardless of your S3 Bucket Key setting, you can include the `x-amz-server-side-encryption-bucket-key-enabled` header with a `true` or `false` value in your request, to override the bucket setting.

Before you configure your bucket to use an S3 Bucket Key, review Changes to note before enabling an S3 Bucket Key.

Configuring an S3 Bucket Key using the Amazon S3 console

When you create a new bucket, you can configure your bucket to use an S3 Bucket Key for SSE-KMS on new objects. You can also configure an existing bucket to use an S3 Bucket Key for SSE-KMS on new objects by updating your bucket properties.

For more information, see Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects.

REST API, AWS CLI, and AWS SDK support for S3 Bucket Keys

You can use the REST API, AWS CLI, or AWS SDK to configure your bucket to use an S3 Bucket Key for SSE-KMS on new objects. You can also enable an S3 Bucket Key at the object level.

For more information, see the following:

- Configuring an S3 Bucket Key at the object level
- Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects

The following API operations support S3 Bucket Keys for SSE-KMS:

- PutBucketEncryption
  - `ServerSideEncryptionRule` accepts the `BucketKeyEnabled` parameter for enabling and disabling an S3 Bucket Key.
- GetBucketEncryption
  - `ServerSideEncryptionRule` returns the settings for `BucketKeyEnabled`.
- PutObject, CopyObject, CreateMultipartUpload, and POST Object
  - The `x-amz-server-side-encryption-bucket-key-enabled` request header enables or disables an S3 Bucket Key at the object level.
- HeadObject, GetObject, UploadPartCopy, UploadPart, and CompleteMultipartUpload
  - The `x-amz-server-side-encryption-bucket-key-enabled` response header indicates if an S3 Bucket Key is enabled or disabled for an object.

Working with CloudFormation

In CloudFormation, the `AWS::S3::Bucket` resource includes an encryption property called `BucketKeyEnabled` that you can use to enable or disable an S3 Bucket Key.

For more information, see Using CloudFormation.

Changes to note before enabling an S3 Bucket Key

Before you enable an S3 Bucket Key, note the following related changes:

IAM or AWS KMS key policies

If your existing AWS Identity and Access Management (IAM) policies or AWS KMS key policies use your object Amazon Resource Name (ARN) as the encryption context to refine or limit access to your KMS key, these policies won't work with an S3 Bucket Key. S3 Bucket Keys use the bucket ARN as encryption context. Before you enable an S3 Bucket Key, update your IAM policies or AWS KMS key policies to use your bucket ARN as the encryption context.

For more information about the encryption context and S3 Bucket Keys, see Encryption context.

CloudTrail events for AWS KMS

After you enable an S3 Bucket Key, your AWS KMS CloudTrail events log your bucket ARN instead of your object ARN. Additionally, you see fewer KMS CloudTrail events for SSE-KMS objects in your logs. Because key material is time-limited in Amazon S3, fewer requests are made to AWS KMS.

Using an S3 Bucket Key with replication

You can use S3 Bucket Keys with Same-Region Replication (SRR) and Cross-Region Replication (CRR).

When Amazon S3 replicates an encrypted object, it generally preserves the encryption settings of the replica object in the destination bucket. However, if the source object is not encrypted and your destination bucket uses default encryption or an S3 Bucket Key, Amazon S3 encrypts the object with the destination bucket's configuration.

The following examples illustrate how an S3 Bucket Key works with replication. For more information, see Replicating encrypted objects (SSE-S3, SSE-KMS, DSSE-KMS, SSE-C).

Example 1 – Source object uses S3 Bucket Keys; destination bucket uses default encryption

If your source object uses an S3 Bucket Key but your destination bucket uses default encryption with SSE-KMS, the replica object maintains its S3 Bucket Key encryption settings in the destination bucket. The destination bucket still uses default encryption with SSE-KMS.

Example 2 – Source object is not encrypted; destination bucket uses an S3 Bucket Key with SSE-KMS

If your source object is not encrypted and the destination bucket uses an S3 Bucket Key with SSE-KMS, the replica object is encrypted by using an S3 Bucket Key with SSE-KMS in the destination bucket. This results in the `ETag` of the source object being different from the `ETag` of the replica object. You must update applications that use the `ETag` to accommodate for this difference.

Working with S3 Bucket Keys

For more information about enabling and working with S3 Bucket Keys, see the following sections:

- Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects
- Configuring an S3 Bucket Key at the object level
- Viewing the settings for an S3 Bucket Key

Configuring your bucket to use an S3 Bucket Key with SSE-KMS for new objects

When you configure server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), you can configure your bucket to use an S3 Bucket Key for SSE-KMS on new objects. S3 Bucket Keys decrease the request traffic from Amazon S3 to AWS KMS and reduce the cost of SSE-KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

You can configure your bucket to use an S3 Bucket Key for SSE-KMS on new objects by using the Amazon S3 console, REST API, AWS SDKs, AWS Command Line Interface (AWS CLI), or CloudFormation. If you want to enable or disable an S3 Bucket Key for existing objects, you can use a `CopyObject` operation. For more information, see Configuring an S3 Bucket Key at the object level and Using Batch Operations to enable S3 Bucket Keys for SSE-KMS.

When an S3 Bucket Key is enabled for the source or destination bucket, the encryption context will be the bucket Amazon Resource Name (ARN) and not the object ARN, for example, `arn:aws:s3:::`*`bucket_ARN`*. You need to update your IAM policies to use the bucket ARN for the encryption context. For more information, see S3 Bucket Keys and replication.

The following examples illustrate how an S3 Bucket Key works with replication. For more information, see Replicating encrypted objects (SSE-S3, SSE-KMS, DSSE-KMS, SSE-C).

Prerequisites

Before you configure your bucket to use an S3 Bucket Key, review Changes to note before enabling an S3 Bucket Key.

Using the S3 console

In the S3 console, you can enable or disable an S3 Bucket Key for a new or existing bucket. Objects in the S3 console inherit their S3 Bucket Key setting from the bucket configuration. When you enable an S3 Bucket Key for your bucket, new objects that you upload to the bucket use an S3 Bucket Key for SSE-KMS.

Uploading, copying, or modifying objects in buckets that have an S3 Bucket Key enabled

If you upload, modify, or copy an object in a bucket that has an S3 Bucket Key enabled, the S3 Bucket Key settings for that object might be updated to align with the bucket configuration.

If an object already has an S3 Bucket Key enabled, the S3 Bucket Key settings for that object don't change when you copy or modify the object. However, if you modify or copy an object that doesn't have an S3 Bucket Key enabled, and the destination bucket has an S3 Bucket Key configuration, the object inherits the destination bucket's S3 Bucket Key settings. For example, if your source object doesn't have an S3 Bucket Key enabled but the destination bucket has S3 Bucket Key enabled, an S3 Bucket Key is enabled for the object.

To enable an S3 Bucket Key when you create a new bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose **Create bucket**.
4. Enter your bucket name, and choose your AWS Region.
5. Under **Default encryption**, for **Encryption key type**, choose **AWS Key Management Service key (SSE-KMS)**.
6. Under **AWS KMS key**, do one of the following to choose your KMS key:

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose your **KMS key** from the list of available keys.

     Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.

- To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

  For more information about creating an AWS KMS key, see Creating Keys in the *AWS Key Management Service Developer Guide*.

7. Under **Bucket Key**, choose **Enable**.
8. Choose **Create bucket**.

   Amazon S3 creates your bucket with an S3 Bucket Key enabled. New objects that you upload to the bucket will use an S3 Bucket Key.

   To disable an S3 Bucket Key, follow the previous steps, and choose **Disable**.

To enable an S3 Bucket Key for an existing bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the bucket that you want to enable an S3 Bucket Key for.
4. Choose the **Properties** tab.
5. Under **Default encryption**, choose **Edit**.
6. Under **Default encryption**, for **Encryption key type**, choose **AWS Key Management Service key (SSE-KMS)**.
7. Under **AWS KMS key**, do one of the following to choose your KMS key:

   - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose your **KMS key** from the list of available keys.

     Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
   - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
   - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

For more information about creating an AWS KMS key, see Creating Keys in the *AWS Key Management Service Developer Guide*.

8. Under **Bucket Key**, choose **Enable**.
9. Choose **Save changes**.

Amazon S3 enables an S3 Bucket Key for new objects added to your bucket. Existing objects don't use the S3 Bucket Key. To configure an S3 Bucket Key for existing objects, you can use a `CopyObject` operation. For more information, see Configuring an S3 Bucket Key at the object level.

To disable an S3 Bucket Key, follow the previous steps, and choose **Disable**.

Using the REST API

You can use PutBucketEncryption to enable or disable an S3 Bucket Key for your bucket. To configure an S3 Bucket Key with `PutBucketEncryption`, use the ServerSideEncryptionRule data type, which includes default encryption with SSE-KMS. You can also optionally use a customer managed key by specifying the KMS key ID for the customer managed key.

For more information and example syntax, see PutBucketEncryption.

Using the AWS SDK for Java

The following example enables default bucket encryption with SSE-KMS and an S3 Bucket Key by using the AWS SDK for Java.

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

ServerSideEncryptionByDefault serverSideEncryptionByDefault = new
 ServerSideEncryptionByDefault()
    .withSSEAlgorithm(SSEAlgorithm.KMS);
ServerSideEncryptionRule rule = new ServerSideEncryptionRule()
    .withApplyServerSideEncryptionByDefault(serverSideEncryptionByDefault)
    .withBucketKeyEnabled(true);
ServerSideEncryptionConfiguration serverSideEncryptionConfiguration =
```

```
    new ServerSideEncryptionConfiguration().withRules(Collections.singleton(rule));

SetBucketEncryptionRequest setBucketEncryptionRequest = new
 SetBucketEncryptionRequest()
    .withServerSideEncryptionConfiguration(serverSideEncryptionConfiguration)
    .withBucketName(bucketName);

s3client.setBucketEncryption(setBucketEncryptionRequest);
```

Using the AWS CLI

The following example enables default bucket encryption with SSE-KMS and an S3 Bucket Key by using the AWS CLI. Replace the *`user input placeholders`* with your own information.

```
aws s3api put-bucket-encryption --bucket amzn-s3-demo-bucket --server-side-encryption-
configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "aws:kms",
                    "KMSMasterKeyID": "KMS-Key-ARN"
                },
                "BucketKeyEnabled": true
            }
        ]
    }'
```

Using CloudFormation

For more information about configuring an S3 Bucket Key with CloudFormation, see AWS::S3::Bucket ServerSideEncryptionRule in the *AWS CloudFormation User Guide*.

Configuring an S3 Bucket Key at the object level

When you perform a PUT or COPY operation using the REST API, AWS SDKs, or AWS CLI, you can enable or disable an S3 Bucket Key at the object level by adding the `x-amz-server-side-encryption-bucket-key-enabled` request header with a `true` or `false` value. S3 Bucket Keys reduce the cost of server-side encryption using AWS Key Management Service (AWS KMS) (SSE-KMS) by decreasing request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

When you configure an S3 Bucket Key for an object using a PUT or COPY operation, Amazon S3 only updates the settings for that object. The S3 Bucket Key settings for the destination bucket do not change. If you submit a PUT or COPY request for a KMS-encrypted object into a bucket with S3 Bucket Keys enabled, your object level operation will automatically use S3 Bucket Keys unless you disable the keys in the request header. If you don't specify an S3 Bucket Key for your object, Amazon S3 applies the S3 Bucket Key settings for the destination bucket to the object.

Prerequisite:

Before you configure your object to use an S3 Bucket Key, review Changes to note before enabling an S3 Bucket Key.

Topics

- Amazon S3 Batch Operations
- Using the REST API
- Using the AWS SDK for Java (PutObject)
- Using the AWS CLI (PutObject)

Amazon S3 Batch Operations

To encrypt your existing Amazon S3 objects, you can use Amazon S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on, and Batch Operations calls the respective API to perform the specified operation.

You can use the S3 Batch Operations Copy operation to copy existing unencrypted objects and write them back to the same bucket as encrypted objects. A single Batch Operations job can perform the specified operation on billions of objects. For more information, see Performing object operations in bulk with Batch Operations and Encrypting objects with Amazon S3 Batch Operations.

Using the REST API

When you use SSE-KMS, you can enable an S3 Bucket Key for an object by using the following API operations:

- PutObject – When you upload an object, you can specify the `x-amz-server-side-encryption-bucket-key-enabled` request header to enable or disable an S3 Bucket Key at the object level.

- CopyObject – When you copy an object and configure SSE-KMS, you can specify the `x-amz-server-side-encryption-bucket-key-enabled` request header to enable or disable an S3 Bucket Key for your object.
- POST Object – When you use a POST operation to upload an object and configure SSE-KMS, you can use the `x-amz-server-side-encryption-bucket-key-enabled` form field to enable or disable an S3 Bucket Key for your object.
- CreateMultipartUpload – When you upload large objects by using the `CreateMultipartUpload` API operation and configure SSE-KMS, you can use the `x-amz-server-side-encryption-bucket-key-enabled` request header to enable or disable an S3 Bucket Key for your object.

To enable an S3 Bucket Key at the object level, include the `x-amz-server-side-encryption-bucket-key-enabled` request header. For more information about SSE-KMS and the REST API, see Using the REST API.

Using the AWS SDK for Java (PutObject)

You can use the following example to configure an S3 Bucket Key at the object level using the AWS SDK for Java.

Java

```
AmazonS3 s3client = AmazonS3ClientBuilder.standard()
    .withRegion(Regions.DEFAULT_REGION)
    .build();

String bucketName = "amzn-s3-demo-bucket1";
String keyName = "key name for object";
String contents = "file contents";

PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName, keyName,
 contents)
    .withBucketKeyEnabled(true);

s3client.putObject(putObjectRequest);
```

Using the AWS CLI (PutObject)

You can use the following AWS CLI example to configure an S3 Bucket Key at the object level as part of a `PutObject` request.

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key object key name --server-side-encryption aws:kms --bucket-key-enabled --body filepath
```

Viewing the settings for an S3 Bucket Key

You can view the settings for an S3 Bucket Key at the bucket or object level by using the Amazon S3 console, REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs.

S3 Bucket Keys decrease request traffic from Amazon S3 to AWS KMS and reduce the cost of server-side encryption using AWS Key Management Service (SSE-KMS). For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.

To view the S3 Bucket Key settings for a bucket or an object that has inherited S3 Bucket Key settings from the bucket configuration, you need permission to perform the `s3:GetEncryptionConfiguration` action. For more information, see GetBucketEncryption in the *Amazon Simple Storage Service API Reference*.

Using the S3 console

In the S3 console, you can view the S3 Bucket Key settings for your bucket or object. S3 Bucket Key settings are inherited from the bucket configuration unless the source objects already has an S3 Bucket Key configured.

Objects and folders in the same bucket can have different S3 Bucket Key settings. For example, if you upload an object using the REST API and enable an S3 Bucket Key for the object, the object retains its S3 Bucket Key setting in the destination bucket, even if S3 Bucket Key is disabled in the destination bucket. As another example, if you enable an S3 Bucket Key for an existing bucket, objects that are already in the bucket do not use an S3 Bucket Key. However, new objects have an S3 Bucket Key enabled.

To view the S3 Bucket Key setting for your bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the bucket that you want to enable an S3 Bucket Key for.
4. Choose **Properties**.
5. In the **Default encryption** section, under **Bucket Key**, you see the S3 Bucket Key setting for your bucket.

   If you can't see the S3 Bucket Key setting, you might not have permission to perform the `s3:GetEncryptionConfiguration` action. For more information, see GetBucketEncryption in the *Amazon Simple Storage Service API Reference*.

To view the S3 Bucket Key setting for your object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the bucket that you want to enable an S3 Bucket Key for.
3. In the **Objects** list, choose your object name.
4. On the **Details** tab, under **Server-side encryption settings**, choose **Edit**.

   Under **Bucket Key**, you see the S3 Bucket Key setting for your object. You cannot edit this setting.

Using the AWS CLI

To return bucket-level S3 Bucket Key settings

To use this example, replace each *user input placeholder* with your own information.

```
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket1
```

For more information, see get-bucket-encryption in the *AWS CLI Command Reference*.

To return object-level S3 Bucket Key settings

To use this example, replace each *user input placeholder* with your own information.

```
aws s3api head-object --bucket amzn-s3-demo-bucket1 --key my_images.tar.bz2
```

For more information, see head-object in the *AWS CLI Command Reference*.

Using the REST API

To return bucket-level S3 Bucket Key settings

To return encryption information for a bucket, including the settings for an S3 Bucket Key, use the `GetBucketEncryption` operation. S3 Bucket Key settings are returned in the response body in the `ServerSideEncryptionConfiguration` element with the `BucketKeyEnabled` setting. For more information, see GetBucketEncryption in the *Amazon S3 API Reference*.

To return object-level settings for an S3 Bucket Key

To return the S3 Bucket Key status for an object, use the `HeadObject` operation. `HeadObject` returns the `x-amz-server-side-encryption-bucket-key-enabled` response header to show whether an S3 Bucket Key is enabled or disabled for the object. For more information, see HeadObject in the *Amazon S3 API Reference*.

The following API operations also return the `x-amz-server-side-encryption-bucket-key-enabled` response header if an S3 Bucket Key is configured for an object:

- PutObject
- PostObject
- CopyObject
- CreateMultipartUpload
- UploadPartCopy
- UploadPart
- CompleteMultipartUpload
- GetObject

Using dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)

Using dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS) applies two layers of encryption to objects when they are uploaded to Amazon S3. DSSE-KMS helps you more easily fulfill compliance standards that require you to apply multilayer encryption to your data and have full control of your encryption keys.

The "dual" in DSSE-KMS refers to two independent layers of AES-256 encryption that are applied to your data:

- *First layer:* Your data is encrypted using a unique data encryption key (DEK) generated by AWS KMS
- *Second layer:* The already-encrypted data is encrypted again using a separate AES-256 encryption key managed by Amazon S3

This differs from standard SSE-KMS, which applies only a single layer of encryption. The dual-layer approach provides enhanced security by ensuring that even if one encryption layer were compromised, your data would remain protected by the second layer. This additional security comes with increased processing overhead and AWS KMS API calls, which accounts for the higher cost compared to standard SSE-KMS. For more information about DSSE-KMS pricing, see AWS KMS key concepts in the AWS Key Management Service Developer Guide and AWS KMS pricing.

When you use DSSE-KMS with an Amazon S3 bucket, the AWS KMS keys must be in the same Region as the bucket. Also, when DSSE-KMS is requested for the object, the S3 checksum that's part of the object's metadata is stored in encrypted form. For more information about checksums, see Checking object integrity in Amazon S3.

> **Note**
>
> S3 Bucket Keys aren't supported for DSSE-KMS.

The key differences between DSSE-KMS and standard SSE-KMS are:

- **Encryption layers:** DSSE-KMS applies two independent layers of AES-256 encryption, while standard SSE-KMS applies one layer
- **Security:** DSSE-KMS provides enhanced protection against potential encryption vulnerabilities
- **Compliance:** DSSE-KMS helps meet regulatory requirements that mandate multilayer encryption
- **Performance:** DSSE-KMS has slightly higher latency due to additional encryption processing
- **Cost:** DSSE-KMS incurs higher charges due to increased computational overhead and additional AWS KMS operations

Requiring dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)

To require dual-layer server-side encryption of all objects in a particular Amazon S3 bucket, you can use a bucket policy. For example, the following bucket policy denies the upload object

(`s3:PutObject`) permission to everyone if the request does not include an `x-amz-server-side-encryption` header that requests server-side encryption with DSSE-KMS.

JSON

```
{
            "Version":"2012-10-17",
            "Id": "PutObjectPolicy",
            "Statement": [{
                  "Sid": "DenyUnEncryptedObjectUploads",
                  "Effect": "Deny",
                  "Principal": {
                      "AWS": "arn:aws:iam::111122223333:root"
                  },
                  "Action": "s3:PutObject",
                  "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
                  "Condition": {
                     "StringNotEquals": {
                        "s3:x-amz-server-side-encryption": "aws:kms:dsse"
                     }
                  }
               }
            ]
         }
```

Topics

- Specifying dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)

Specifying dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)

You can apply encryption when you are either uploading a new object or copying an existing object.

You can specify DSSE-KMS by using the Amazon S3 console, Amazon S3 REST API, and the AWS Command Line Interface (AWS CLI). For more information, see the following topics.

> **Note**
>
> You can use multi-Region AWS KMS keys in Amazon S3. However, Amazon S3 currently treats multi-Region keys as though they were single-Region keys, and does not use the

multi-Region features of the key. For more information, see Using multi-Region keys in the *AWS Key Management Service Developer Guide*.

Note

If you want to use a KMS key that is owned by a different account, you must have permission to use the key. For more information about cross-account permissions for KMS keys, see Creating KMS keys that other accounts can use in the *AWS Key Management Service Developer Guide*.

Using the S3 console

This section describes how to set or change the type of encryption of an object to use dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS) by using the Amazon S3 console.

Note

- You can change an object's encryption if your object is less than 5 GB. If your object is greater than 5 GB, you must use the AWS CLI or AWS SDKs to change an object's encryption.
- For a list of additional permissions required to change an object's encryption, see the section called "Required permissions for S3 API operations". For example policies that grant this permission, see the section called "Identity-based policy examples".
- If you change an object's encryption, a new object is created to replace the old one. If S3 Versioning is enabled, a new version of the object is created, and the existing object becomes an older version. The role that changes the property also becomes the owner of the new object (or object version).

To add or change encryption for an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the navigation pane, choose **Buckets**, and then choose the **General purpose buckets** tab. Navigate to the Amazon S3 bucket or folder that contains the objects you want to change.
3. Select the check box for the objects you want to change.
4. On the **Actions** menu, choose **Edit server-side encryption** from the list of options that appears.
5. Scroll to the **Server-side encryption** section.
6. Under **Encryption settings**, choose **Use bucket settings for default encryption** or **Override bucket settings for default encryption**.
7. If you chose **Override bucket settings for default encryption**, configure the following encryption settings.

    a. Under **Encryption type**, choose **Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)**.

    b. Under **AWS KMS key**, do one of the following to choose your KMS key:

    - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and then choose your **KMS key** from the list of available keys.

      Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
    - To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and then enter your KMS key ARN in the field that appears.
    - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

      For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

> **Important**
>
> You can use only KMS keys that are available in the same AWS Region as the bucket. The Amazon S3 console lists only the first 100 KMS keys in the same Region as the bucket. To use a KMS key that is not listed, you must enter your KMS key ARN. If you want to use a KMS key that is owned by a different account, you

must first have permission to use the key, and then you must enter the KMS key ARN.

Amazon S3 supports only symmetric encryption KMS keys, and not asymmetric KMS keys. For more information, see Identifying asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

8. For **Bucket Key**, choose **Disable**. S3 Bucket Keys aren't supported for DSSE-KMS.
9. Under **Additional copy settings**, choose whether you want to **Copy source settings**, **Don't specify settings**, or **Specify settings**. **Copy source settings** is the default option. If you only want to copy the object without the source settings attributes, choose **Don't specify settings**. Choose **Specify settings** to specify settings for storage class, ACLs, object tags, metadata, server-side encryption, and additional checksums.
10. Choose **Save changes**.

Note

This action applies encryption to all specified objects. When you're encrypting folders, wait for the save operation to finish before adding new objects to the folder.

Using the REST API

When you create an object—that is, when you upload a new object or copy an existing object—you can specify the use of dual-layer server-side encryption with AWS KMS keys (DSSE-KMS) to encrypt your data. To do this, add the `x-amz-server-side-encryption` header to the request. Set the value of the header to the encryption algorithm `aws:kms:dsse`. Amazon S3 confirms that your object is stored with DSSE-KMS encryption by returning the response header `x-amz-server-side-encryption`.

If you specify the `x-amz-server-side-encryption` header with a value of `aws:kms:dsse`, you can also use the following request headers:

- `x-amz-server-side-encryption-aws-kms-key-id: SSEKMSKeyId`
- `x-amz-server-side-encryption-context: SSEKMSEncryptionContext`

Topics

- Amazon S3 REST API operations that support DSSE-KMS
- Encryption context (x-amz-server-side-encryption-context)
- AWS KMS key ID (x-amz-server-side-encryption-aws-kms-key-id)

Amazon S3 REST API operations that support DSSE-KMS

The following REST API operations accept the `x-amz-server-side-encryption`, `x-amz-server-side-encryption-aws-kms-key-id`, and `x-amz-server-side-encryption-context` request headers.

- PutObject – When you upload data by using the PUT API operation, you can specify these request headers.
- CopyObject – When you copy an object, you have both a source object and a target object. When you pass DSSE-KMS headers with the `CopyObject` operation, they are applied only to the target object. When you're copying an existing object, regardless of whether the source object is encrypted or not, the destination object is not encrypted unless you explicitly request server-side encryption.
- POST Object – When you use a POST operation to upload an object, instead of the request headers, you provide the same information in the form fields.
- CreateMultipartUpload – When you upload large objects by using a multipart upload, you can specify these headers in the `CreateMultipartUpload` request.

The response headers of the following REST API operations return the `x-amz-server-side-encryption` header when an object is stored with server-side encryption.

- PutObject
- CopyObject
- POST Object
- CreateMultipartUpload
- UploadPart
- UploadPartCopy
- CompleteMultipartUpload
- GetObject
- HeadObject

Important

- All GET and PUT requests for an object that's protected by AWS KMS fail if you don't make them by using Secure Sockets Layer (SSL), Transport Layer Security (TLS), or Signature Version 4.
- If your object uses DSSE-KMS, don't send encryption request headers for GET requests and HEAD requests, or you'll get an HTTP 400 (Bad Request) error.

Encryption context (`x-amz-server-side-encryption-context`)

If you specify `x-amz-server-side-encryption:aws:kms:dsse`, the Amazon S3 API supports an encryption context with the `x-amz-server-side-encryption-context` header. An encryption context is a set of key-value pairs that contain additional contextual information about the data.

Amazon S3 automatically uses the object's Amazon Resource Name (ARN) as the encryption context pair; for example, `arn:aws:s3:::`*`object_ARN`*.

You can optionally provide an additional encryption context pair by using the `x-amz-server-side-encryption-context` header. However, because the encryption context is not encrypted, make sure it does not include sensitive information. Amazon S3 stores this additional key pair alongside the default encryption context.

For information about the encryption context in Amazon S3, see Encryption context. For general information about the encryption context, see AWS Key Management Service Concepts - Encryption context in the *AWS Key Management Service Developer Guide*.

AWS KMS key ID (`x-amz-server-side-encryption-aws-kms-key-id`)

You can use the `x-amz-server-side-encryption-aws-kms-key-id` header to specify the ID of the customer managed key that's used to protect the data. If you specify the `x-amz-server-side-encryption:aws:kms:dsse` header but don't provide the `x-amz-server-side-encryption-aws-kms-key-id` header, Amazon S3 uses the AWS managed key (`aws/s3`) to protect the data. If you want to use a customer managed key, you must provide the `x-amz-server-side-encryption-aws-kms-key-id` header of the customer managed key.

> **Important**
>
> When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys. For more information about these keys, see Symmetric encryption KMS keys in the *AWS Key Management Service Developer Guide*.

Using the AWS CLI

When you upload a new object or copy an existing object, you can specify the use of DSSE-KMS to encrypt your data. To do this, add the `--server-side-encryption aws:kms:dsse` parameter to the request. Use the `--ssekms-key-id` *`example-key-id`* parameter to add your customer managed AWS KMS key that you created. If you specify `--server-side-encryption aws:kms:dsse`, but do not provide an AWS KMS key ID, then Amazon S3 will use the AWS managed key (`aws/s3`).

```
aws s3api put-object --bucket amzn-s3-demo-bucket --key example-object-key --server-side-encryption aws:kms:dsse --ssekms-key-id example-key-id --body filepath
```

You can encrypt an unencrypted object to use DSSE-KMS by copying the object back in place.

```
aws s3api copy-object --bucket amzn-s3-demo-bucket --key example-object-key --copy-source amzn-s3-demo-bucket/example-object-key --server-side-encryption aws:kms:dsse --ssekms-key-id example-key-id
```

Using server-side encryption with customer-provided keys (SSE-C)

Server-side encryption is about protecting data at rest. Server-side encryption encrypts only the object data, not the object metadata. You can use server-side encryption with customer-provided keys (SSE-C) in your general purpose buckets to encrypt your data with your own encryption keys. With the encryption key that you provide as part of your request, Amazon S3 manages data encryption as it writes to disks and data decryption when you access your objects. Therefore, you don't need to maintain any code to perform data encryption and decryption. The only thing that you need to do is manage the encryption keys that you provide.

Starting April 2026, SSE-C is disabled by default for all new general purpose buckets and existing buckets in accounts with no SSE-C encrypted objects. Most modern workloads use server-side encryption with Amazon S3 managed keys (SSE-S3) or AWS KMS keys (SSE-KMS) instead, because

SSE-C requires you to provide the encryption key with every request, making it impractical to share access with other users, roles, or AWS services that operate on your data. To learn more about SSE-KMS, see the section called "KMS keys stored in AWS KMS (SSE-KMS)".

If your workload requires SSE-C, you must explicitly enable it by setting `BlockedEncryptionTypes` to `NONE` in your bucket's default encryption configuration using the `PutBucketEncryption` API. While SSE-C is blocked, any `PutObject`, `CopyObject`, `PostObject`, Multipart Upload, or replication request that specifies SSE-C encryption is rejected with an HTTP 403 `AccessDenied` error. To learn more, see the section called "Blocking or unblocking SSE-C".

There are no additional charges for using SSE-C. However, requests to configure and use SSE-C incur standard Amazon S3 request charges. For information about pricing, see Amazon S3 pricing.

> **Important**
>
> Amazon Simple Storage Service now applies a new default bucket security setting that automatically disables server-side encryption with customer-provided keys (SSE-C) for all new general purpose buckets. In April 2026, Amazon S3 deployed an update so all new general purpose buckets have SSE-C encryption disabled for all new write requests. For existing buckets in AWS accounts with no SSE-C encrypted objects, Amazon S3 also disabled SSE-C for all new write requests. With this change, applications that need SSE-C encryption must deliberately enable SSE-C by using the PutBucketEncryption API operation after creating a new bucket. For more information about this change, see Default SSE-C setting for new buckets FAQ.

Considerations before using SSE-C

- S3 never stores the encryption key when you use SSE-C. You must supply the encryption key every time you want anyone to download your SSE-C encrypted data from S3.
  - You manage a mapping of which encryption key was used to encrypt which object. You are responsible for tracking which encryption key you provided for which object. That also means if you lose the encryption key, you lose the object.
  - Because you manage encryption keys on the client side, you manage any additional safeguards, such as key rotation, on the client side.
  - This design can make it difficult to share your SSE-C key with other users, roles, or AWS services that need to operate on your data. Due to the widespread support for SSE-KMS across

AWS, most modern workloads do not use SSE-C because it lacks the flexibility of SSE-KMS. To learn more about SSE-KMS, see Using server-side encryption with AWS KMS keys (SSE-KMS).

  - This means that objects encrypted with SSE-C cannot be natively decrypted by AWS managed services.

- You must use HTTPS when specifying SSE-C headers on your requests.
  - Amazon S3 rejects any requests made over HTTP when using SSE-C. For security considerations, we recommend that you consider any key that you erroneously send over HTTP to be compromised. Discard the key and rotate as appropriate.
- If your bucket is versioning-enabled, each object version that you upload can have its own encryption key. You are responsible for tracking which encryption key was used for which object version.
- SSE-C is not supported in the Amazon S3 Console. You cannot use the Amazon S3 console to upload an object and specify SSE-C encryption. You also cannot use the console to update (for example, change the storage class or add metadata) an existing object stored using SSE-C.
- SSE-C is blocked by default for new buckets. You must explicitly enable SSE-C using the `PutBucketEncryption` API before you can upload objects with SSE-C encryption. For more information, see the section called "Blocking or unblocking SSE-C".

Topics

- Specifying server-side encryption with customer-provided keys (SSE-C)
- Blocking or unblocking SSE-C for a general purpose bucket
- Default SSE-C setting for new buckets FAQ

Specifying server-side encryption with customer-provided keys (SSE-C)

Server-side encryption with customer-provided keys (SSE-C) is disabled by default for new general purpose buckets. Before you can upload objects using SSE-C, you must enable it by setting `BlockedEncryptionTypes` to `NONE` in your bucket's default encryption configuration. For more information, see Blocking or unblocking SSE-C for a general purpose bucket.

Once SSE-C is enabled for your bucket, you can use SSE-C in your upload requests by passing the required headers. See Amazon S3 actions that support writing data with SSE-C, and make sure to include the S3 API headers required for SSE-C object encryption and decryption requests.

When you upload an object specifying SSE-C, Amazon S3 uses the encryption key that you provide to apply AES-256 encryption to your data. Amazon S3 then removes the encryption key from memory. When you retrieve an object, you must provide the same encryption key as part of your request. Amazon S3 first verifies that the encryption key that you provided matches, and then it decrypts the object before returning the object data to you.

Before using SSE-C, make sure you have reviewed the Considerations before using SSE-C.

> **Note**
>
> Amazon S3 does not store the encryption key that you provide. Instead, it stores a randomly salted Hash-based Message Authentication Code (HMAC) value of the encryption key to validate future requests. The salted HMAC value cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object. That means if you lose the encryption key, you lose the object.

Topics

- SSE-C Actions and Required Headers
- Example bucket policy to enforce SSE-C encryption
- Presigned URLs and SSE-C
- Making requests with SSE-C
- Using the REST API
- Using the AWS SDKs to specify SSE-C for PUT, GET, Head, and Copy operations
- Using the AWS SDKs to specify SSE-C for multipart uploads

SSE-C Actions and Required Headers

Specifying SSE-C on supported S3 APIs requires passing specific request parameters.

> **Note**
>
> The `PutBucketEncryption` API does not support setting SSE-C as a default encryption method for a bucket, objects won't automatically use SSE-C. SSE-C is an object-level encryption method where you provide the encryption key with each upload or download request. Amazon S3 uses this key to encrypt or decrypt the object during the request and then discards the key.

However, you do use `PutBucketEncryption` to control whether SSE-C uploads are allowed for the bucket via the `BlockedEncryptionTypes` parameter. By default, new buckets have SSE-C blocked, you must set `BlockedEncryptionTypes` to `NONE` to permit SSE-C uploads.

Amazon S3 actions that support writing data with SSE-C

You can request server-side encryption with customer provided keys (SSE-C) when writing objects to a general purpose bucket by using the following API operations or actions:

- CopyObject
- CreateMultipartUpload
- CompleteMultipartUpload
- POST Object
- PutObject
- UploadPart
- UploadPartCopy

Note

S3 Replication supports objects that are encrypted with SSE-C. For more information about replicating encrypted objects, see Replicating encrypted objects (SSE-S3, SSE-KMS, DSSE-KMS, SSE-C).

S3 API headers required for SSE-C object encryption and decryption requests

You must provide the following three API headers to encrypt or decrypt objects with SSE-C:

- `x-amz-server-side-encryption-customer-algorithm` Use this header to specify the encryption algorithm. The header value must be AES256.
- `x-amz-server-side-encryption-customer-key` Use this header to provide the 256-bit, base64-encoded encryption key for Amazon S3 to use to encrypt or decrypt your data.
- `x-amz-server-side-encryption-customer-key-MD5` Use this header to provide the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321. Amazon S3

uses this header for a message integrity check to ensure that the encryption key was transmitted without error.

S3 API headers required for requests to copy source objects encrypted with SSE-C

You must provide the following three API headers to copy source objects encrypted with SSE-C:

- `x-amz-copy-source-server-side-encryption-customer-algorithm` Include this header to specify the algorithm Amazon S3 should use to decrypt the source object. This value must be AES256.
- `x-amz-copy-source-server-side-encryption-customer-key` Include this header to provide the base64-encoded encryption key for Amazon S3 to use to decrypt the source object. This encryption key must be the one that you provided Amazon S3 when you created the source object. Otherwise, Amazon S3 cannot decrypt the object.
- `x-amz-copy-source-server-side-encryption-customer-key-MD5` Include this header to provide the base64-encoded 128-bit MD5 digest of the encryption key according to RFC 1321.

Example bucket policy to enforce SSE-C encryption

To require SSE-C for all objects written to an Amazon S3 bucket, you can use a bucket policy. For example, the following bucket policy denies upload object (`s3:PutObject`) permissions for all requests that don't include the `x-amz-server-side-encryption-customer-algorithm` header requesting SSE-C.

```
{
"Version":"2012-10-17",
    "Id": "PutObjectPolicy",
    "Statement": [
        {
"Sid": "RequireSSECObjectUploads",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
            "Null": {
              "s3:x-amz-server-side-encryption-customer-algorithm": "true"
                }
```

```
            }
        }
    ]
}
```

Important

If you use a bucket policy to require SSE-C on `s3:PutObject`, you must include the `x-amz-server-side-encryption-customer-algorithm` header in all multipart upload requests (CreateMultipartUpload, UploadPart, and CompleteMultipartUpload).

Presigned URLs and SSE-C

You can generate a presigned URL that can be used for operations such as uploading a new object, retrieving an existing object, or retrieving object metadata. Presigned URLs support SSE-C as follows:

- When creating a presigned URL, you must specify the algorithm by using the `x-amz-server-side-encryption-customer-algorithm` header in the signature calculation.
- When using the presigned URL to upload a new object, retrieve an existing object, or retrieve only object metadata, you must provide all the encryption headers in your client application's request.

Note

For non-SSE-C objects, you can generate a presigned URL and directly paste that URL into a browser to access the data.
However, you cannot do this for SSE-C objects, because in addition to the presigned URL, you also must include HTTP headers that are specific to SSE-C objects. Therefore, you can use presigned URLs for SSE-C objects only programmatically.

For more information about presigned URLs, see Download and upload objects with presigned URLs.

Making requests with SSE-C

At the time of object creation with the REST API, you can specify server-side encryption with customer-provided keys (SSE-C). When you use SSE-C, you must provide encryption key information using the S3 API headers required for requests to copy source objects encrypted with SSE-C. You can use AWS SDK wrapper libraries to add these headers to your request. If you need to, you can make the Amazon S3 REST API calls directly in your application.

> **Important**
>
> SSE-C is disabled by default for all new buckets. Before uploading objects with SSE-C, you must enable it by updating your bucket's default encryption configuration. For more information, see Blocking or unblocking SSE-C for a general purpose bucket.

> **Note**
>
> You cannot use the Amazon S3 console to upload an object and request SSE-C. You also cannot use the console to update (for example, change the storage class or add metadata) an existing object stored using SSE-C. For more information, see S3 API headers required for SSE-C object encryption and decryption requests.

Using the REST API

Amazon S3 REST APIs that support SSE-C

The following Amazon S3 APIs support server-side encryption with customer-provided encryption keys (SSE-C).

- **GET operation** – When retrieving objects using the GET API (see GET Object), you can specify the request headers.
- **HEAD operation** – To retrieve object metadata using the HEAD API (see HEAD Object), you can specify these request headers.
- **PUT operation** – When uploading data using the PUT Object API (see PUT Object), you can specify these request headers.
- **Multipart Upload** – When uploading large objects using the multipart upload API, you can specify these headers. You specify these headers in the initiate request (see Initiate Multipart

Upload) and each subsequent part upload request (see Upload Part or UploadPartCopy). For each part upload request, the encryption information must be the same as what you provided in the initiate multipart upload request.

- **POST operation** – When using a POST operation to upload an object (see POST Object), instead of the request headers, you provide the same information in the form fields.
- **Copy operation** – When you copy an object (see CopyObject), you have both a source object and a target object:
  - If you want to specify the target object's encryption type, you must provide the `x-amz-server-side-encryption` request header.
  - If you want the target object encrypted using SSE-C, you must provide encryption information using the S3 API S3 API headers required for SSE-C object encryption and decryption requests.
  - If the source object is encrypted using SSE-C, you must provide encryption key information using the S3 API headers S3 API headers required for requests to copy source objects encrypted with SSE-C.

Using the AWS SDKs to specify SSE-C for PUT, GET, Head, and Copy operations

The following examples show how to request server-side encryption with customer-provided keys (SSE-C) for objects. The examples perform the following operations. Each operation shows how to specify SSE-C-related headers in the request:

- **Put object** – Uploads an object and requests server-side encryption using a customer-provided encryption key.
- **Get object** – Downloads the object uploaded in the previous step. In the request, you provide the same encryption information you provided when you uploaded the object. Amazon S3 needs this information to decrypt the object so that it can return it to you.
- **Get object metadata** – Retrieves the object's metadata. You provide the same encryption information used when the object was created.
- **Copy object** – Makes a copy of the previously-uploaded object. Because the source object is stored using SSE-C, you must provide its encryption information in your copy request. By default, Amazon S3 encrypts the copy of the object only if you explicitly request it. This example directs Amazon S3 to store an encrypted copy of the object.

Java

> **Note**
>
> This example shows how to upload an object in a single operation. When using the Multipart Upload API to upload large objects, you provide encryption information in the same way shown in this example. For examples of multipart uploads that use the AWS SDK for Java, see Uploading an object using multipart upload.

To add the required encryption information, you include an `SSECustomerKey` in your request. For more information about the `SSECustomerKey` class, see the REST API section.

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.*;

import javax.crypto.KeyGenerator;
import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

public class ServerSideEncryptionUsingClientSideEncryptionKey {
    private static SSECustomerKey SSE_KEY;
    private static AmazonS3 S3_CLIENT;
    private static KeyGenerator KEY_GENERATOR;

    public static void main(String[] args) throws IOException,
 NoSuchAlgorithmException {
```

```
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String keyName = "*** Key name ***";
        String uploadFileName = "*** File path ***";
        String targetKeyName = "*** Target key name ***";

        // Create an encryption key.
        KEY_GENERATOR = KeyGenerator.getInstance("AES");
        KEY_GENERATOR.init(256, new SecureRandom());
        SSE_KEY = new SSECustomerKey(KEY_GENERATOR.generateKey());

        try {
            S3_CLIENT = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Upload an object.
            uploadObject(bucketName, keyName, new File(uploadFileName));

            // Download the object.
            downloadObject(bucketName, keyName);

            // Verify that the object is properly encrypted by attempting to
retrieve it
            // using the encryption key.
            retrieveObjectMetadata(bucketName, keyName);

            // Copy the object into a new object that also uses SSE-C.
            copyObject(bucketName, keyName, targetKeyName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    private static void uploadObject(String bucketName, String keyName, File file) {
        PutObjectRequest putRequest = new PutObjectRequest(bucketName, keyName,
file).withSSECustomerKey(SSE_KEY);
```

```
        S3_CLIENT.putObject(putRequest);
        System.out.println("Object uploaded");
    }

    private static void downloadObject(String bucketName, String keyName) throws
IOException {
        GetObjectRequest getObjectRequest = new GetObjectRequest(bucketName,
keyName).withSSECustomerKey(SSE_KEY);
        S3Object object = S3_CLIENT.getObject(getObjectRequest);

        System.out.println("Object content: ");
        displayTextInputStream(object.getObjectContent());
    }

    private static void retrieveObjectMetadata(String bucketName, String keyName) {
        GetObjectMetadataRequest getMetadataRequest = new
GetObjectMetadataRequest(bucketName, keyName)
                .withSSECustomerKey(SSE_KEY);
        ObjectMetadata objectMetadata =
S3_CLIENT.getObjectMetadata(getMetadataRequest);
        System.out.println("Metadata retrieved. Object size: " +
objectMetadata.getContentLength());
    }

    private static void copyObject(String bucketName, String keyName, String
targetKeyName)
            throws NoSuchAlgorithmException {
        // Create a new encryption key for target so that the target is saved using
        // SSE-C.
        SSECustomerKey newSSEKey = new SSECustomerKey(KEY_GENERATOR.generateKey());

        CopyObjectRequest copyRequest = new CopyObjectRequest(bucketName, keyName,
bucketName, targetKeyName)
                .withSourceSSECustomerKey(SSE_KEY)
                .withDestinationSSECustomerKey(newSSEKey);

        S3_CLIENT.copyObject(copyRequest);
        System.out.println("Object copied");
    }

    private static void displayTextInputStream(S3ObjectInputStream input) throws
IOException {
        // Read one line at a time from the input stream and display each line.
        BufferedReader reader = new BufferedReader(new InputStreamReader(input));
```

```
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        System.out.println();
    }
}
```

.NET

> **Note**
>
> For examples of uploading large objects using the multipart upload API, see Uploading an object using multipart upload and Using the AWS SDKs (low-level API).

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSEClientEncryptionKeyObjectOperationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string keyName = "*** key name for new object created ***";
        private const string copyTargetKeyName = "*** key name for object copy ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
```

```
        {
            client = new AmazonS3Client(bucketRegion);
            ObjectOpsUsingClientEncryptionKeyAsync().Wait();
        }
        private static async Task ObjectOpsUsingClientEncryptionKeyAsync()
        {
            try
            {
                // Create an encryption key.
                Aes aesEncryption = Aes.Create();
                aesEncryption.KeySize = 256;
                aesEncryption.GenerateKey();
                string base64Key = Convert.ToBase64String(aesEncryption.Key);

                // 1. Upload the object.
                PutObjectRequest putObjectRequest = await
UploadObjectAsync(base64Key);
                // 2. Download the object and verify that its contents matches what
you uploaded.
                await DownloadObjectAsync(base64Key, putObjectRequest);
                // 3. Get object metadata and verify that the object uses AES-256
encryption.
                await GetObjectMetadataAsync(base64Key);
                // 4. Copy both the source and target objects using server-side
encryption with
                //    a customer-provided encryption key.
                await CopyObjectAsync(aesEncryption, base64Key);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        private static async Task<PutObjectRequest> UploadObjectAsync(string
base64Key)
        {
            PutObjectRequest putObjectRequest = new PutObjectRequest
```

```
            {
                BucketName = bucketName,
                Key = keyName,
                ContentBody = "sample text",
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };
            PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
            return putObjectRequest;
        }
        private static async Task DownloadObjectAsync(string base64Key,
PutObjectRequest putObjectRequest)
        {
            GetObjectRequest getObjectRequest = new GetObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                // Provide encryption information for the object stored in Amazon
S3.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (String.Compare(putObjectRequest.ContentBody, content) == 0)
                    Console.WriteLine("Object content is same as we uploaded");
                else
                    Console.WriteLine("Error...Object content is not same.");

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                    Console.WriteLine("Object encryption method is AES256, same as
we set");
                else
                    Console.WriteLine("Error...Object encryption method is not the
same as AES256 we set");
```

```
                // Assert.AreEqual(putObjectRequest.ContentBody, content);
                // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getResponse.ServerSideEncryptionCustomerMethod);
            }
        }
        private static async Task GetObjectMetadataAsync(string base64Key)
        {
            GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
            {
                BucketName = bucketName,
                Key = keyName,

                // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };

            GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
            Console.WriteLine("The object metadata show encryption method used is:
{0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
            // Assert.AreEqual(ServerSideEncryptionCustomerMethod.AES256,
getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
        }
        private static async Task CopyObjectAsync(Aes aesEncryption, string
base64Key)
        {
            aesEncryption.GenerateKey();
            string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

            CopyObjectRequest copyRequest = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = keyName,
                DestinationBucket = bucketName,
                DestinationKey = copyTargetKeyName,
                // Information about the source object's encryption.
                CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,
```

```
                // Information about the target object's encryption.
                ServerSideEncryptionCustomerMethod =
 ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = copyBase64Key
            };
            await client.CopyObjectAsync(copyRequest);
        }
    }
}
```

Using the AWS SDKs to specify SSE-C for multipart uploads

The example in the preceding section shows how to request server-side encryption with customer-provided key (SSE-C) in the PUT, GET, Head, and Copy operations. This section describes other Amazon S3 APIs that support SSE-C.

Java

To upload large objects, you can use multipart upload APIs. For more information, see Uploading and copying objects using multipart upload in Amazon S3. You can use either high-level or low-level APIs to upload large objects. These APIs support encryption-related headers in the request.

- When using the high-level `TransferManager` API, you provide the encryption-specific headers in the `PutObjectRequest`. For more information, see Uploading an object using multipart upload.
- When using the low-level API, you provide encryption-related information in the `InitiateMultipartUploadRequest`, followed by identical encryption information in each `UploadPartRequest`. You do not need to provide any encryption-specific headers in your `CompleteMultipartUploadRequest`. For examples, see Using the AWS SDKs (low-level API).

The following example uses `TransferManager` to create objects and shows how to provide SSE-C related information. The example does the following:

- Creates an object using the `TransferManager.upload()` method. In the `PutObjectRequest` instance, you provide encryption key information in the request. Amazon S3 encrypts the object using the customer-provided key.

- Makes a copy of the object by calling the `TransferManager.copy()` method. The example directs Amazon S3 to encrypt the object copy using a new `SSECustomerKey`. Because the source object is encrypted using SSE-C, the `CopyObjectRequest` also provides the encryption key of the source object so that Amazon S3 can decrypt the object before copying it.

Example

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.CopyObjectRequest;
import com.amazonaws.services.s3.model.PutObjectRequest;
import com.amazonaws.services.s3.model.SSECustomerKey;
import com.amazonaws.services.s3.transfer.Copy;
import com.amazonaws.services.s3.transfer.TransferManager;
import com.amazonaws.services.s3.transfer.TransferManagerBuilder;
import com.amazonaws.services.s3.transfer.Upload;

import javax.crypto.KeyGenerator;
import java.io.File;
import java.security.SecureRandom;

public class ServerSideEncryptionCopyObjectUsingHLwithSSEC {

    public static void main(String[] args) throws Exception {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";
        String fileToUpload = "*** File path ***";
        String keyName = "*** New object key name ***";
        String targetKeyName = "*** Key name for object copy ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withRegion(clientRegion)
                    .withCredentials(new ProfileCredentialsProvider())
                    .build();
            TransferManager tm = TransferManagerBuilder.standard()
```

```
                    .withS3Client(s3Client)
                    .build();

            // Create an object from a file.
            PutObjectRequest putObjectRequest = new PutObjectRequest(bucketName,
keyName, new File(fileToUpload));

            // Create an encryption key.
            KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");
            keyGenerator.init(256, new SecureRandom());
            SSECustomerKey sseCustomerEncryptionKey = new
SSECustomerKey(keyGenerator.generateKey());

            // Upload the object. TransferManager uploads asynchronously, so this
call
            // returns immediately.
            putObjectRequest.setSSECustomerKey(sseCustomerEncryptionKey);
            Upload upload = tm.upload(putObjectRequest);

            // Optionally, wait for the upload to finish before continuing.
            upload.waitForCompletion();
            System.out.println("Object created.");

            // Copy the object and store the copy using SSE-C with a new key.
            CopyObjectRequest copyObjectRequest = new CopyObjectRequest(bucketName,
keyName, bucketName, targetKeyName);
            SSECustomerKey sseTargetObjectEncryptionKey = new
SSECustomerKey(keyGenerator.generateKey());
            copyObjectRequest.setSourceSSECustomerKey(sseCustomerEncryptionKey);

copyObjectRequest.setDestinationSSECustomerKey(sseTargetObjectEncryptionKey);

            // Copy the object. TransferManager copies asynchronously, so this call
returns
            // immediately.
            Copy copy = tm.copy(copyObjectRequest);

            // Optionally, wait for the upload to finish before continuing.
            copy.waitForCompletion();
            System.out.println("Copy complete.");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
```

```
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

To upload large objects, you can use multipart upload API (see Uploading and copying objects using multipart upload in Amazon S3). AWS SDK for .NET provides both high-level or low-level APIs to upload large objects. These APIs support encryption-related headers in the request.

- When using high-level `Transfer-Utility` API, you provide the encryption-specific headers in the `TransferUtilityUploadRequest` as shown. For code examples, see Uploading an object using multipart upload.

  ```
  TransferUtilityUploadRequest request = new TransferUtilityUploadRequest()
  {
      FilePath = filePath,
      BucketName = existingBucketName,
      Key = keyName,
      // Provide encryption information.
      ServerSideEncryptionCustomerMethod =
   ServerSideEncryptionCustomerMethod.AES256,
      ServerSideEncryptionCustomerProvidedKey = base64Key,
  };
  ```

- When using the low-level API, you provide encryption-related information in the initiate multipart upload request, followed by identical encryption information in the subsequent upload part requests. You do not need to provide any encryption-specific headers in your complete multipart upload request. For examples, see Using the AWS SDKs (low-level API).

  The following is a low-level multipart upload example that makes a copy of an existing large object. In the example, the object to be copied is stored in Amazon S3 using SSE-C, and you want to save the target object also using SSE-C. In the example, you do the following:

  - Initiate a multipart upload request by providing an encryption key and related information.
  - Provide source and target object encryption keys and related information in the `CopyPartRequest`.

- Obtain the size of the source object to be copied by retrieving the object metadata.
- Upload the objects in 5 MB parts.

Example

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class SSECLowLevelMPUcopyObjectTest
    {
        private const string existingBucketName = "*** bucket name ***";
        private const string sourceKeyName      = "*** source object key name
 ***";
        private const string targetKeyName      = "*** key name for the target
 object ***";
        private const string filePath           = @"*** file path ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CopyObjClientEncryptionKeyAsync().Wait();
        }

        private static async Task CopyObjClientEncryptionKeyAsync()
        {
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);

            await CreateSampleObjUsingClientEncryptionKeyAsync(base64Key,
 s3Client);
```

```
            await CopyObjectAsync(s3Client, base64Key);
        }
        private static async Task CopyObjectAsync(IAmazonS3 s3Client, string
base64Key)
        {
            List<CopyPartResponse> uploadResponses = new List<CopyPartResponse>();

            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = targetKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key,
            };

            InitiateMultipartUploadResponse initResponse =
                await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB
            long firstByte = 0;
            long lastByte = partSize;

            try
            {
                // First find source object size. Because object is stored
encrypted with
                // customer provided key you need to provide encryption
information in your request.
                GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest()
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                    ServerSideEncryptionCustomerProvidedKey = base64Key // " *
**source object encryption key ***"
                };
```

```
                GetObjectMetadataResponse getObjectMetadataResponse = await
s3Client.GetObjectMetadataAsync(getObjectMetadataRequest);

                long filePosition = 0;
                for (int i = 1; filePosition <
getObjectMetadataResponse.ContentLength; i++)
                {
                    CopyPartRequest copyPartRequest = new CopyPartRequest
                    {
                        UploadId = initResponse.UploadId,
                        // Source.
                        SourceBucket = existingBucketName,
                        SourceKey = sourceKeyName,
                        // Source object is stored using SSE-C. Provide encryption
information.
                        CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        CopySourceServerSideEncryptionCustomerProvidedKey =
base64Key, //"***source object encryption key ***",
                        FirstByte = firstByte,
                        // If the last part is smaller then our normal part size
then use the remaining size.
                        LastByte = lastByte >
getObjectMetadataResponse.ContentLength ?
                            getObjectMetadataResponse.ContentLength - 1 :
lastByte,

                        // Target.
                        DestinationBucket = existingBucketName,
                        DestinationKey = targetKeyName,
                        PartNumber = i,
                        // Encryption information for the target object.
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key
                    };
                    uploadResponses.Add(await
s3Client.CopyPartAsync(copyPartRequest));
                    filePosition += partSize;
                    firstByte += partSize;
                    lastByte += partSize;
                }

                // Step 3: complete.
```

```
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId,
                };
                completeRequest.AddPartETags(uploadResponses);

                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);
            }
            catch (Exception exception)
            {
                Console.WriteLine("Exception occurred: {0}", exception.Message);
                AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = targetKeyName,
                    UploadId = initResponse.UploadId
                };
                s3Client.AbortMultipartUpload(abortMPURequest);
            }
        }
        private static async Task
CreateSampleObjUsingClientEncryptionKeyAsync(string base64Key, IAmazonS3
s3Client)
        {
            // List to store upload part responses.
            List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

            // 1. Initialize.
            InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest
            {
                BucketName = existingBucketName,
                Key = sourceKeyName,
                ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                ServerSideEncryptionCustomerProvidedKey = base64Key
            };
```

```
            InitiateMultipartUploadResponse initResponse =
               await s3Client.InitiateMultipartUploadAsync(initiateRequest);

            // 2. Upload Parts.
            long contentLength = new FileInfo(filePath).Length;
            long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

            try
            {
                long filePosition = 0;
                for (int i = 1; filePosition < contentLength; i++)
                {
                    UploadPartRequest uploadRequest = new UploadPartRequest
                    {
                        BucketName = existingBucketName,
                        Key = sourceKeyName,
                        UploadId = initResponse.UploadId,
                        PartNumber = i,
                        PartSize = partSize,
                        FilePosition = filePosition,
                        FilePath = filePath,
                        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
                        ServerSideEncryptionCustomerProvidedKey = base64Key
                    };

                    // Upload part and add response to our list.
                    uploadResponses.Add(await
s3Client.UploadPartAsync(uploadRequest));

                    filePosition += partSize;
                }

                // Step 3: complete.
                CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId,
                    //PartETags = new List<PartETag>(uploadResponses)

                };
                completeRequest.AddPartETags(uploadResponses);
```

```
                CompleteMultipartUploadResponse completeUploadResponse =
                    await s3Client.CompleteMultipartUploadAsync(completeRequest);

            }
            catch (Exception exception)
            {
                Console.WriteLine("Exception occurred: {0}", exception.Message);
                AbortMultipartUploadRequest abortMPURequest = new
 AbortMultipartUploadRequest
                {
                    BucketName = existingBucketName,
                    Key = sourceKeyName,
                    UploadId = initResponse.UploadId
                };
                await s3Client.AbortMultipartUploadAsync(abortMPURequest);
            }
        }
    }
}
```

Blocking or unblocking SSE-C for a general purpose bucket

Starting April 2026, Amazon S3 automatically disables server-side encryption with customer-provided keys (SSE-C) for all new general purpose buckets. Amazon S3 also disabled SSE-C for existing buckets in accounts with no SSE-C encrypted objects. This means that by default, requests to upload objects using SSE-C are rejected with an HTTP 403 `AccessDenied` error.

SSE-C requires you to provide the encryption key with every request to read or write encrypted objects, making it difficult to share access with other users, roles, or AWS services that operate on your data. Most workloads use server-side encryption with Amazon S3 managed keys (SSE-S3) or AWS KMS keys (SSE-KMS) instead.

If your workload requires SSE-C, you can explicitly enable it by updating the default encryption configuration for your bucket. Conversely, if you have existing buckets where SSE-C is still allowed, you can block it to prevent new SSE-C uploads.

When SSE-C is blocked for a bucket, any `PutObject`, `CopyObject`, `PostObject`, Multipart Upload, or replication request that specifies SSE-C encryption will be rejected with an HTTP 403 `AccessDenied` error. Existing SSE-C encrypted objects in the bucket are unaffected, you can still read them with `GetObject` or `HeadObject` by providing the required SSE-C headers.

This setting is a parameter on the `PutBucketEncryption` API and can also be updated using the S3 console, AWS CLI, or AWS SDKs. You must have the `s3:PutEncryptionConfiguration` permission.

> **Important**
>
> Amazon Simple Storage Service now applies a new default bucket security setting that automatically disables server-side encryption with customer-provided keys (SSE-C) for all new general purpose buckets. In April 2026, Amazon S3 deployed an update so all new general purpose buckets have SSE-C encryption disabled for all new write requests. For existing buckets in AWS accounts with no SSE-C encrypted objects, Amazon S3 also disabled SSE-C for all new write requests. With this change, applications that need SSE-C encryption must deliberately enable SSE-C by using the PutBucketEncryption API operation after creating a new bucket. For more information about this change, see Default SSE-C setting for new buckets FAQ.

Permissions

Use the `PutBucketEncryption` API or the S3 Console, AWS SDKs, or AWS CLI to block or unblock encryption types for a general purpose bucket. You must have the following permission:

- `s3:PutEncryptionConfiguration`

Use the `GetBucketEncryption` API or the S3 Console, AWS SDKs, or AWS CLI to view blocked encryption types for a general purpose bucket. You must have the following permission:

- `s3:GetEncryptionConfiguration`

Considerations before blocking SSE-C encryption

After you block SSE-C for any bucket, the following encryption behavior applies:

- There is no change to the encryption of the objects that existed in the bucket before you blocked SSE-C encryption.
- After you block SSE-C encryption, you can continue to make GetObject and HeadObject requests on pre-existing objects encrypted with SSE-C as long as you provide the required SSE-C headers on the requests.

- When SSE-C is blocked for a bucket, any `PutObject`, `CopyObject`, `PostObject`, or Multipart Upload requests that specify SSE-C encryption will be rejected with an HTTP 403 `AccessDenied` error.
- If a destination bucket for replication has SSE-C blocked and the source objects being replicated are encrypted with SSE-C, the replication will fail with an HTTP 403 `AccessDenied` error.

If you want to review if you're using SSE-C encryption in any of your buckets before blocking this encryption type, you can use tools such as AWS CloudTrail to monitor access to your data. This blog post shows you how to audit encryption methods for object uploads in real time. You can also reference this re:Post article to guide you through the querying S3 Inventory reports to see if you have any SSE-C encrypted objects.

Steps

You can block or unblock server-side encryption with customer-provided keys (SSE-C) for a general purpose bucket by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To block or unblock SSE-C encryption for a bucket using the Amazon S3 console:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **general purpose buckets**.
3. Select the bucket that you would like to block SSE-C encryption for.
4. Select the **Properties** tab for the bucket.
5. Navigate to the **Default Encryption** properties panel for the bucket and select **Edit**.
6. In the **Blocked encryption types** section, check the box next to **Server-side encryption with customer-provided keys (SSE-C)** to block SSE-C encryption or uncheck this box to allow SSE-C.
7. Select **Save Changes**.

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to block or unblock SSE-C encryption for a general purpose bucket by using the AWS CLI. To use the command replace the *user input placeholders* with your own information.

Request to block SSE-C encryption for a general purpose bucket:

```
aws s3api put-bucket-encryption \
  --bucket amzn-s3-demo-bucket \
  --server-side-encryption-configuration '{
    "Rules": [{
      "BlockedEncryptionTypes": {
        "EncryptionType": ["SSE-C"]
      }
    }]
  }'
```

Request to enable the use of SSE-C encryption on a general purpose bucket:

```
aws s3api put-bucket-encryption \
  --bucket amzn-s3-demo-bucket \
  --server-side-encryption-configuration '{
    "Rules": [{
      "BlockedEncryptionTypes": {
        "EncryptionType": ["NONE"]
      }
    }]
  }'
```

Using the AWS SDKs

SDK for Java 2.x

The following examples show you how to block or unblock SSE-C encryption writes to your general purpose buckets by using the AWS SDKs

Example - PutBucketEncryption request setting the default encryption configuration to SSE-S3 and blocking SSE-C

```
S3Client s3Client = ...;
ServerSideEncryptionByDefault defaultSse = ServerSideEncryptionByDefault
        .builder()
```

```
        .sseAlgorithm(ServerSideEncryption.AES256)
        .build();
BlockedEncryptionTypes blockedEncryptionTypes = BlockedEncryptionTypes
        .builder()
        .encryptionType(EncryptionType.SSE_C)
        .build();
ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
        .applyServerSideEncryptionByDefault(defaultSse)
        .blockedEncryptionTypes(blockedEncryptionTypes)
        .build();
s3Client.putBucketEncryption(be -> be
        .bucket(bucketName)
        .serverSideEncryptionConfiguration(c -> c.rules(rule)));
```

Example - PutBucketEncryption request setting the default encryption configuration to SSE-S3 and unblocking SSE-C

```
S3Client s3Client = ...;
ServerSideEncryptionByDefault defaultSse = ServerSideEncryptionByDefault
        .builder()
        .sseAlgorithm(ServerSideEncryption.AES256)
        .build();
BlockedEncryptionTypes blockedEncryptionTypes = BlockedEncryptionTypes
        .builder()
        .encryptionType(EncryptionType.NONE)
        .build();
ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
        .applyServerSideEncryptionByDefault(defaultSse)
        .blockedEncryptionTypes(blockedEncryptionTypes)
        .build();
s3Client.putBucketEncryption(be -> be
        .bucket(bucketName)
        .serverSideEncryptionConfiguration(c -> c.rules(rule)));
```

SDK for Python (Boto3)

Example - PutBucketEncryption request setting the default encryption configuration to SSE-S3 and blocking SSE-C

```
s3 = boto3.client("s3")
s3.put_bucket_encryption(
    Bucket="amzn-s3-demo-bucket",
```

```
    ServerSideEncryptionConfiguration={
        "Rules":[{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            },
            "BlockedEncryptionTypes": {
                "EncryptionType": ["SSE-C"]
            }
        }]
    }
)
```

Example - PutBucketEncryption request setting the default encryption configuration to SSE-S3 and unblocking SSE-C

```
s3 = boto3.client("s3")
s3.put_bucket_encryption(
    Bucket="amzn-s3-demo-bucket",
    ServerSideEncryptionConfiguration={
        "Rules":[{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            },
            "BlockedEncryptionTypes": {
                "EncryptionType": ["NONE"]
            }
        }]
    }
)
```

Using the REST API

For information about the Amazon S3 REST API support for blocking or unblocking SSE-C encryption for a general purpose bucket, see the following section in the Amazon Simple Storage Service API Reference:

- BlockedEncryptionTypes data type used in the ServerSideEncryptionRule data type of the PutBucketEncryption and GetBucketEncryption API operations.

Default SSE-C setting for new buckets FAQ

> **Important**
>
> Amazon Simple Storage Service now applies a new default bucket security setting that automatically disables server-side encryption with customer-provided keys (SSE-C) for all new general purpose buckets. In April 2026, Amazon S3 deployed an update so all new general purpose buckets have SSE-C encryption disabled for all new write requests. For existing buckets in AWS accounts with no SSE-C encrypted objects, Amazon S3 also disabled SSE-C for all new write requests. With this change, applications that need SSE-C encryption must deliberately enable SSE-C by using the PutBucketEncryption API operation after creating a new bucket.

The following sections answer questions about this update.

1. Does the new SSE-C setting take effect for all newly created buckets?

Yes. This deployment completed in 37 AWS Regions, including the AWS China and AWS GovCloud (US) Regions, in April 2026.

> **Note**
>
> All newly created buckets in all AWS Regions except Middle East (Bahrain) and Middle East (UAE) will have SSE-C disabled by default.

2. Did Amazon S3 update my existing bucket configurations?

If your AWS account did not have any SSE-C encrypted objects, then AWS disabled SSE-C encryption on all of your existing buckets. If any bucket in your AWS account had SSE-C encrypted objects, then AWS did not change the bucket configurations on any of your buckets in that account. The new default setting applies to all new general purpose buckets.

3. Can I disable SSE-C encryption for my buckets?

Yes. You can disable SSE-C encryption for any bucket by calling the PutBucketEncryption API operation and specifying the new `BlockedEncryptionTypes` header.

4. Can I use SSE-C to encrypt data in my new buckets?

Yes. Most modern use cases in Amazon S3 no longer use SSE-C because it lacks the flexibility of server-side encryption is with Amazon S3 managed keys (SSE-S3) or server-side encryption with AWS KMS keys (SSE-KMS). If you need to use SSE-C encryption in a new bucket, you can create the new bucket and then enable the use of SSE-C encryption in a separate `PutBucketEncryption` request.

Example

```
aws s3api create-bucket \
bucket amzn-s3-demo-bucket \
region us-east-1 \

aws s3api put-bucket-encryption \
-- bucket amzn-s3-demo-bucket \
-- server-side-encryption-configuration \
'{ \Rules\: [{
   {
   \ApplyServerSideEncryptionByDefault\: {
     \SSEAlgorithm\: \AES256\,
    },
   \BlockedEncryptionTypes\: [
     \EncryptionType\:\NONE\]
   }
   }]
}'
```

Note

You must have the `s3:PutEncryptionConfiguration`permission to call the `PutBucketEncryption` API.

5. How does blocking SSE-C affect requests to my bucket?

When SSE-C is blocked for a bucket, any `PutObject`, `CopyObject`, `PostObject`, or Multipart Upload or replication requests that specify SSE-C encryption will be rejected with an HTTP 403 `AccessDenied` error.

Protecting data by using client-side encryption

Client-side encryption is the act of encrypting your data locally to help ensure its security in transit and at rest. To encrypt your objects before you send them to Amazon S3, use the Amazon S3 Encryption Client. When your objects are encrypted in this manner, your objects aren't exposed to any third party, including AWS. Amazon S3 receives your objects already encrypted; Amazon S3 does not play a role in encrypting or decrypting your objects. You can use both the Amazon S3 Encryption Client and server-side encryption to encrypt your data. When you send encrypted objects to Amazon S3, Amazon S3 doesn't recognize the objects as being encrypted, it only detects typical objects.

The Amazon S3 Encryption Client works as an intermediary between you and Amazon S3. After you instantiate the Amazon S3 Encryption Client, your objects are automatically encrypted and decrypted as part of your Amazon S3 `PutObject` and `GetObject` requests. Your objects are all encrypted with a unique data key. The Amazon S3 Encryption Client does not use or interact with bucket keys, even if you specify a KMS key as your wrapping key.

The *Amazon S3 Encryption Client Developer Guide* focuses on versions 3.0 and later of the Amazon S3 Encryption Client. For more information, see What is the Amazon S3 Encryption Client? in the *Amazon S3 Encryption Client Developer Guide*.

For more information about previous versions of the Amazon S3 Encryption client, see the AWS SDK Developer Guide for your programming language.

- AWS SDK for Java
- AWS SDK for .NET
- AWS SDK for Go
- AWS SDK for PHP
- AWS SDK for Ruby
- AWS SDK for C++

Protecting data in transit with encryption

Amazon S3 supports both HTTP and HTTPS protocols for data transmission. HTTP transmits data in plain text, while HTTPS adds a security layer by encrypting data using Transport Layer Security (TLS). TLS protects against eavesdropping, data tampering, and man-in-the-middle attacks. While

HTTP traffic is accepted, most implementations use encryption in transit with HTTPS and TLS to protect data as it travels between clients and Amazon S3.

TLS 1.2 and TLS 1.3 Support

Amazon S3 supports TLS 1.2 and TLS 1.3 for HTTPS connections across all API endpoints for all AWS Regions. S3 automatically negotiates the strongest TLS protection supported by your client software, and the S3 endpoint you are accessing. Current AWS tools (2014 or later) including the AWS SDKs and AWS CLI automatically default to TLS 1.3 with no action required on your part. You can override this automatic negotiation through client configuration settings to specify a particular TLS version if backward compatibility to TLS 1.2 is needed. When using TLS 1.3, you can optionally configure hybrid post quantum key exchange (ML-KEM) to make quantum-resistant requests to Amazon S3. For more information, see the section called “Configure PQ-TLS client”.

> **Note**
>
> TLS 1.3 is supported in all S3 endpoints, except for AWS PrivateLink for Amazon S3 and Multi-Region Access Points.

Monitoring TLS usage

You can use either Amazon S3 server access logs or AWS CloudTrail to monitor requests to Amazon S3 buckets. Both logging options record the TLS version and cipher suite used in each request.

- **Amazon S3 server access logs** – Server access logging provides detailed records for the requests that are made to a bucket. For example, access log information can be useful in security and access audits. For more information, see Amazon S3 server access log format.
- **AWS CloudTrail** – AWS CloudTrail is a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures all API calls for Amazon S3 as events. For more information, see Amazon S3 CloudTrail events.

Enforcing encryption in transit

It’s a security best practice to enforce encryption of data in transit to Amazon S3. You can enforce HTTPS-only communication or the use of specific TLS version through various policy mechanisms. These include IAM resource-based policies for S3 buckets (bucket policies), Service Control Policies (SCPs), Resource Control Policies (RCPs), and VPC endpoint policies.

Bucket policy examples for enforcing encryption in transit

You can use the S3 condition key `s3:TlsVersion` to restrict access to Amazon S3 buckets based on the TLS version that's used by the client. For more information, see Example 6: Requiring a minimum TLS version.

Example bucket policy enforcing TLS 1.3 using the `S3:TlsVersion` condition key

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyInsecureConnections",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1",
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ],
      "Condition": {
        "NumericLessThan": {
          "s3:TlsVersion": "1.3"
        }
      }
    }
  ]
}
```

You can use the `aws:SecureTransport` global condition key in your S3 bucket policy to check whether the request was sent through HTTPS (TLS). Unlike the previous example, this condition does not check for a specific TLS version. For more information, see Restrict access to only HTTPS requests.

Example bucket policy enforcing HTTPS using the `aws:SecureTransport` global condition key

```
{
    "Version":"2012-10-17",
    "Statement": [
     {
        "Sid": "RestrictToTLSRequestsOnly",
        "Action": "s3:*",
```

```
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket1",
            "arn:aws:s3:::amzn-s3-demo-bucket1/*"
        ],
        "Condition": {
            "Bool": {
                "aws:SecureTransport": "false"
            }
        },
        "Principal": "*"
    }
  ]
}
```

Example policy based on both keys and more examples

You can use both types of condition keys in the previous examples in one policy. For more information and additional enforcement approaches, see the AWS Storage Blog article Enforcing encryption in transit with TLS1.2 or higher with Amazon S3.

Using hybrid post-quantum TLS with Amazon S3

Amazon S3 supports a hybrid post-quantum key exchange option for the TLS network encryption protocol. You can use this TLS option when you make requests to Amazon S3 endpoints utilizing TLS 1.3. The classic cipher suites that S3 supports for TLS sessions make brute force attacks on the key exchange mechanisms infeasible with current technology. However, if a cryptographically relevant quantum computer becomes practical in the future, the classic cipher suites used in TLS key exchange mechanisms will be susceptible to these attacks. At present, the industry is aligned on hybrid post-quantum key exchange that combines classic and post-quantum elements to ensure that your TLS connection is at least as strong as it would be with classic cipher suites. Amazon S3 supports hybrid PQ-TLS, in compliance with the industry-standard IANA specification, today.

If you're developing applications that rely on the long-term confidentiality of data passed over a TLS connection, you should consider a plan to migrate to post-quantum cryptography before large-scale quantum computers become available for use. As part of the shared responsibility model, S3 enables quantum-safe cryptography on our service endpoints. As browsers and applications enable PQ-TLS on their side, S3 will choose the strongest possible configuration to secure data in transit.

Supported endpoint types and AWS Regions

Post-quantum TLS for Amazon S3 is available in all AWS Regions. For a list of S3 endpoints for each AWS Region, see Amazon Simple Storage Service endpoints and quotas in the *Amazon Web Services General Reference*.

> **Note**
>
> Hybrid post-quantum TLS is supported for all S3 endpoints except for AWS PrivateLink for Amazon S3, Multi-Region Access Points, and S3 Vectors.

Using hybrid post-quantum TLS with Amazon S3

You must configure the client that makes requests to Amazon S3 to support hybrid post-quantum TLS. When setting up your HTTP client test environment or production environments, be aware of the following information:

Encryption in Transit

Hybrid post-quantum TLS is only used for encryption in transit. This protects your data while it is traveling from your client to the S3 endpoint. This new support combined with Amazon S3's server-side encryption by default utilizing AES-256 algorithms offers customers quantum-resistant encryption both in-transit and at-rest. For more information about server-side encryption in Amazon S3, see Protecting data with server-side encryption.

Supported Clients

The use of hybrid post-quantum TLS requires using a client that supports this functionality. AWS SDKs and tools have cryptographic capabilities and configuration that differ across languages and runtimes. To learn more about post-quantum cryptography for specific tools, see Enabling hybrid post-quantum TLS.

> **Note**
>
> PQ-TLS key exchange details for requests to Amazon S3 are not available in AWS CloudTrail events or S3 server access logs.

Learn more about post-quantum TLS

For more information about using hybrid post-quantum TLS, see the following resources.

- To learn about post-quantum cryptography at AWS, including links to blog posts and research papers, see Post-Quantum Cryptography for AWS.
- For information about s2n-tls, see Introducing s2n-tls, a New Open Source TLS Implementation and Using s2n-tls.
- For information about the AWS Common Runtime HTTP Client, see Configuring the AWS CRT-based HTTP client in the *AWS SDK for Java 2.x Developer Guide*.
- For information about the post-quantum cryptography project at the National Institute for Standards and Technology (NIST), see Post-Quantum Cryptography.
- For information about NIST post-quantum cryptography standardization, see NIST's Post-Quantum Cryptography Standardization.

Configuring hybrid post-quantum TLS for your client

To use PQ-TLS with Amazon S3, you need to configure your client to support post-quantum key exchange algorithms. Also ensure that your client supports the hybrid approach, which combines traditional elliptic curve cryptography with post-quantum algorithms like ML-KEM (Module-Lattice-Based Key Encapsulation Mechanism).

The specific configuration depends on your client library and programming language. For more information, see Enabling hybrid post-quantum TLS.

Client configuration example: AWS SDK for Java 2

In this procedure, add a Maven dependency for the AWS Common Runtime HTTP Client. Next, configure an HTTP client that prefers post-quantum TLS. Then, create an Amazon S3 client that uses the HTTP client.

> **Note**
>
> The AWS Common Runtime HTTP Client, which has been available as a preview, became generally available in February 2023. In that release, the `tlsCipherPreference` class and the `tlsCipherPreference()` method parameter are replaced by the `postQuantumTlsEnabled()` method parameter. If you were using this example during the preview, you need to update your code.

1. Add the AWS Common Runtime client to your Maven dependencies. We recommend using the latest available version.

For example, this statement adds version `2.30.22` of the AWS Common Runtime client to your Maven dependencies.

```
<dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>aws-crt-client</artifactId>
    <version>2.30.22</version>
</dependency>
```

2. To enable the hybrid post-quantum cipher suites, add the AWS SDK for Java 2.x to your project and initialize it. Then enable the hybrid post-quantum cipher suites on your HTTP client as shown in the following example.

   This code uses the `postQuantumTlsEnabled()` method parameter to configure an AWS common runtime HTTP client that prefers the recommended hybrid post-quantum cipher suite, ECDH with ML-KEM. Then it uses the configured HTTP client to build an instance of the Amazon S3 asynchronous client, `S3AsyncClient`. After this code completes, all Amazon S3 API requests on the `S3AsyncClient` instance use hybrid post-quantum TLS.

   > **Important**
   >
   > As of v2.35.11, callers no longer need to set `.postQuantumTlsEnabled(true)` to enable hybrid post-quantum TLS for your client. All versions newer than v2.35.11 enable post-quantum TLS by default.

   ```
   // Configure HTTP client
   SdkAsyncHttpClient awsCrtHttpClient = AwsCrtAsyncHttpClient.builder()
              .postQuantumTlsEnabled(true)
              .build();

   // Create the Amazon S3 async client
   S3AsyncClient s3Async = S3AsyncClient.builder()
             .httpClient(awsCrtHttpClient)
             .build();
   ```

3. Test your Amazon S3 calls with hybrid post-quantum TLS.

When you call Amazon S3 API operations on the configured Amazon S3 client, your calls are transmitted to the Amazon S3 endpoint using hybrid post-quantum TLS. To test your configuration, call an Amazon S3 API, such as `ListBuckets`.

```
ListBucketsResponse reponse = s3Async.listBuckets();
```

Test your hybrid post-quantum TLS configuration

Consider running the following tests with hybrid cipher suites on your applications that call Amazon S3.

- Run load tests and benchmarks. The hybrid cipher suites perform differently than traditional key exchange algorithms. You might need to adjust your connection timeouts to allow for the longer handshake times. If you're running inside an AWS Lambda function, extend the execution timeout setting.
- Try connecting from different locations. Depending on the network path your request takes, you might discover that intermediate hosts, proxies, or firewalls with deep packet inspection (DPI) block the request. This might result from using the new cipher suites in the ClientHello part of the TLS handshake, or from the larger key exchange messages. If you have trouble resolving these issues, work with your security team or IT administrators to update the relevant configuration and unblock the new TLS cipher suites.

Internetwork traffic privacy

This topic describes how Amazon S3 secures connections from the service to other locations.

Traffic between service and on-premises clients and applications

The following connections can be combined with AWS PrivateLink to provide connectivity between your private network and AWS:

- An AWS Site-to-Site VPN connection. For more information, see What is AWS Site-to-Site VPN?
- An Direct Connect connection. For more information, see What is Direct Connect?

Access to Amazon S3 through the network is through AWS published APIs. Clients must support Transport Layer Security (TLS) 1.2. We recommend utilizing TLS 1.3 with hybrid post-quantum

key exchange. Clients must also support cipher suites with Perfect Forward Secrecy (PFS), such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). Most modern systems such as Java 7 and later support these modes. Additionally, you must sign requests using an access key ID and a secret access key that are associated with an IAM principal, or you can use the AWS Security Token Service (STS) to generate temporary security credentials to sign requests.

Traffic between AWS resources in the same Region

A virtual private cloud (VPC) endpoint for Amazon S3 is a logical entity within a VPC that allows connectivity only to Amazon S3. The VPC routes requests to Amazon S3 and routes responses back to the VPC. For more information, see VPC Endpoints in the *VPC User Guide*. For example bucket policies that you can use to control S3 bucket access from VPC endpoints, see Controlling access from VPC endpoints with bucket policies.

AWS PrivateLink for Amazon S3

With AWS PrivateLink for Amazon S3, you can provision *interface VPC endpoints* (interface endpoints) in your virtual private cloud (VPC). These endpoints are directly accessible from applications that are on premises over VPN and Direct Connect, or in a different AWS Region over VPC peering.

Interface endpoints are represented by one or more elastic network interfaces (ENIs) that are assigned private IP addresses from subnets in your VPC. Requests to Amazon S3 over interface endpoints stay on the Amazon network. You can also access interface endpoints in your VPC from on-premises applications through AWS Direct Connect or AWS Virtual Private Network (Site-to-Site VPN). For more information about how to connect your VPC with your on-premises network, see the Direct Connect User Guide and the AWS Site-to-Site VPN User Guide.

For general information about interface endpoints, see Interface VPC endpoints (AWS PrivateLink) in the *AWS PrivateLink Guide*.

Topics

- Types of VPC endpoints for Amazon S3
- Restrictions and limitations of AWS PrivateLink for Amazon S3
- Creating a VPC endpoint
- Accessing Amazon S3 interface endpoints
- IP Address types for VPC endpoints

- DNS record IP types for VPC endpoints
- Private DNS
- Accessing buckets, access points, and Amazon S3 Control API operations from S3 interface endpoints
- Updating an on-premises DNS configuration
- Creating a VPC endpoint policy for Amazon S3

Types of VPC endpoints for Amazon S3

You can use two types of VPC endpoints to access Amazon S3: *gateway endpoints* and *interface endpoints* (by using AWS PrivateLink). A *gateway endpoint* is a gateway that you specify in your route table to access Amazon S3 from your VPC over the AWS network. *Interface endpoints* extend the functionality of gateway endpoints by using private IP addresses to route requests to Amazon S3 from within your VPC, on premises, or from a VPC in another AWS Region by using VPC peering or AWS Transit Gateway. For more information, see What is VPC peering? and Transit Gateway vs VPC peering.

Interface endpoints are compatible with gateway endpoints. If you have an existing gateway endpoint in the VPC, you can use both types of endpoints in the same VPC.

Gateway endpoints for Amazon S3	Interface endpoints for Amazon S3
In both cases, your network traffic remains on the AWS network.	
Use Amazon S3 public IP addresses	Use private IP addresses from your VPC to access Amazon S3
Use the same Amazon S3 DNS names	Require endpoint-specific Amazon S3 DNS names
Do not allow access from on premises	Allow access from on premises
Do not allow access from another AWS Region	Allow access from a VPC in another AWS Region by using VPC peering or AWS Transit Gateway
Not billed	Billed

For more information, see Gateway endpoints and interface VPC endpoints in the *AWS PrivateLink Guide*.

Restrictions and limitations of AWS PrivateLink for Amazon S3

VPC limitations apply to AWS PrivateLink for Amazon S3. For more information, see Interface endpoint considerations and AWS PrivateLink quotas in the *AWS PrivateLink Guide*. In addition, the following restrictions apply.

Interface endpoints for Amazon S3 does not support the following:

- Website endpoints
- Legacy global endpoints
- S3 dash Region endpoints
- Using CopyObject or UploadPartCopy between buckets in different AWS Regions
- Transport Layer Security (TLS) 1.0
- Transport Layer Security (TLS) 1.1
- Transport Layer Security (TLS) 1.3
- Hybrid post-quantum Transport Layer Security (TLS)

Creating a VPC endpoint

To create a VPC interface endpoint, see Create a VPC endpoint in the *AWS PrivateLink Guide*. To create a VPC gateway endpoint, see Create a gateway endpoint in the *AWS PrivateLink Guide*.

FIPS endpoints for Amazon S3

To connect to Amazon S3 over AWS PrivateLink with FIPS 140-3 validated cryptographic modules, create an interface endpoint. Use the FIPS service name `com.amazonaws.region.s3-fips`. For example, use `com.amazonaws.us-east-1.s3-fips`. FIPS interface endpoints are available in all AWS Regions that offer FIPS endpoints for Amazon S3. These endpoints support access to buckets, access points, and Amazon S3 Control API operations. You can configure FIPS interface endpoints to use IPv4, IPv6, or dual-stack IP addressing.

Accessing Amazon S3 interface endpoints

When you create an interface endpoint, Amazon S3 generates two types of endpoint-specific, S3 DNS names: *Regional* and *zonal*.

- A *Regional* DNS name includes a unique VPC endpoint ID, a service identifier, the AWS Region, and `vpce.amazonaws.com` in its name. For example, for VPC endpoint ID *vpce-1a2b3c4d*, the DNS name generated might be similar to *vpce-1a2b3c4d-5e6f*`.s3.us-east-1.vpce.amazonaws.com`.
- A *Zonal* DNS name includes the Availability Zone—for example, *vpce-1a2b3c4d-5e6f*`-us-east-1a.s3.us-east-1.vpce.amazonaws.com`. You might use this option if your architecture isolates Availability Zones. For example, you could use it for fault containment or to reduce Regional data transfer costs.

Endpoint-specific S3 DNS names can be resolved from the S3 public DNS domain.

If you create an interface endpoint that uses the FIPS service name `com.amazonaws.`*region*`.s3-fips`, Amazon S3 generates endpoint-specific DNS names. These names include the `s3-fips` service identifier. For example, *vpce-1a2b3c4d-5e6f*`.s3-fips.us-east-1.vpce.amazonaws.com`.

VPC endpoints for Amazon S3 support different types of IP addressing, including: IPv4, IPv6, and Dualstack. See IP Address types for VPC endpoints and DNS record IP types for VPC endpoints.

IP Address types for VPC endpoints

VPC endpoints for Amazon S3 support different types of IP addressing, including:

- **IPv4**

  VPC endpoints can be configured to use only IPv4 addresses for connectivity.

- **IPv6**

  VPC endpoints can be configured to use only IPv6 addresses for connectivity.

- **Dual-stack**

  VPC endpoints can be configured in dual-stack mode, supporting both IPv4 and IPv6 addresses simultaneously. This provides flexibility to access Amazon S3 over either IPv4 or IPv6 networks.

The IP address type you choose for your VPC endpoint will depend on the networking requirements of your applications and infrastructure. Considerations may include the IP addressing schemes used in your VPC, on-premises networks, and across the internet connectivity to Amazon S3. For more information, see IP address types for interface endpoints and gateway endpoints in the *Amazon Virtual Private Cloud* guide.

DNS record IP types for VPC endpoints

Depending on your IP address type, when you call a VPC endpoint, the AWS service can return A records, AAAA records, or both A and AAAA records. You can customize which record types your AWS service returns by modifying the DNS record IP type. The following table shows the supported DNS record IP types and the IP address types:

Supported IP address types	DNS record IP types
IPv4	IPv4
IPv6	IPv6
Dualstack	Dualstack, IPv4, IPv6, Service-defined

Configuring the service-defined DNS record IP type for Amazon S3

If you create a gateway endpoint for Amazon S3 and configure the DNS record IP type as service-defined and use the regional service endpoint,(such as `s3.us-east-2.amazonaws.com`), Amazon S3 returns A records to your clients. In contrast, if you create a gateway endpoint and are using a dual-stack service endpoint, (such as `s3.dualstack.us-east-2.amazonaws.com`) and select the DNS record IP type as `service-defined`, Amazon S3 returns both A and AAAA records to your clients.

Similarly, if you create an interface endpoint with Private DNS enabled and choose service-defined as the DNS record type, for the regional service endpoint, (such as, `s3.us-east-2.amazonaws.com`), Amazon S3 returns A records to your client. Whereas, for a dualstack service endpoint, (such as, `s3.dualstack.us-east-2.amazonaws.com`), Amazon S3 returns both A and AAAA records. For more information, see DNS record IP type for interface endpoints and gateway endpoints in the *VPC User Guide*.

The following table shows the supported DNS record IP types for gateway and interface endpoints:

IP address type	Supported DNS record IP types for S3 Gateway endpoints	Supported DNS record IP types for S3 Interface endpoints
IPv4	IPv4, service-defined*	IPv4
IPv6	IPv6, service-defined*	IPv6
Dualstack	IPv4, IPv6, Dualstack, service-defined*	Dualstack*, IPv4, IPv6, service-defined

* Represents the default DNS record IP type.

To enable IPv6 connectivity on an existing S3 gateway or interface endpoint, update the IP address type for the endpoint to **`Dualstack`**. When updated, Amazon S3 automatically updates the routing tables with IPv6 addresses for gateway endpoints. Then you can use the dualstack service endpoint, such as `s3.dualstack.us-east-2.amazonaws.com`, and Amazon S3 will return both `A` and `AAAA` records for dualstack S3 DNS queries. If you want to use IPV6 with the regional service endpoint, such as `s3.us-east-2.amazonaws.com`, modify the IP address type for the endpoint to **`Dualstack`**; and the DNS record IP type to **`Dualstack`**. Then Amazon S3 will return both `A` and `AAAA` records for the regional S3 DNS queries.

Considerations

- If your gateway endpoint has the default configuration of IP address type as `IPv4` and DNS record IP type as `service-defined`, then the dualstack service endpoint (such as `s3.dualstack.us-east-2.amazonaws.com`), traffic that uses `AAAA` records won't be routed through the gateway endpoint. Instead, this traffic will be dropped or routed over an IPv6-compatible path if one is present. For example, if your virtual private cloud (VPC) has an internet gateway, your IPv6 traffic will be routed over the internet gateway in your VPC in this scenario. If you want to ensure your traffic is always routed over a VPC endpoint, you can use an Amazon S3 bucket policy that restricts access to a specific bucket if a specified VPC endpoint is not used. For more information, see Restricting access to a specific VPC endpoint.
- If your interface endpoint has the default configuration of IP address type, which is IPv4, and DNS record IP type as IPv4, dualstack service endpoints, such as

ass3.dualstack.us-east-2.amazonaws.com, are not supported. Traffic that uses the A or AAAA records of the dualstack service endpoints won't be routed through the interface endpoint. Instead, this traffic will be dropped or routed over another compatible path if one is present.

- Creating or modifying a gateway endpoint with a DNS record IP type other than service-defined requires both the enableDnsSupport and enableDnsHostnames VPC attributes to be set to true.

Private DNS

Private DNS options for VPC interface endpoints simplify routing S3 traffic over VPC endpoints and help you take advantage of the lowest-cost network path available to your application. You can use private DNS options to route Regional S3 traffic without updating your S3 clients to use the endpoint-specific DNS names of your interface endpoints, or managing DNS infrastructure. With private DNS names enabled, Regional S3 DNS queries resolve to the private IP addresses of AWS PrivateLink for the following endpoints:

- Regional bucket endpoints (for example, s3.us-east-1.amazonaws.com, or s3-fips.us-east-1.amazonaws.com for FIPS)
- Control endpoints (for example, s3-control.us-east-1.amazonaws.com, or s3-control-fips.us-east-1.amazonaws.com for FIPS)
- Access point endpoints (for example, s3-accesspoint.us-east-1.amazonaws.com, or s3-accesspoint-fips.us-east-1.amazonaws.com for FIPS)

If you configure the interface endpoint with the **Dualstack** IP address type, the FIPS dual-stack endpoints also resolve to the private IP addresses of AWS PrivateLink. For example, s3-fips.dualstack.us-east-1.amazonaws.com resolves to a private IP address.

If you have a gateway endpoint in your VPC, you can automatically route in-VPC requests over your existing S3 gateway endpoint and on-premises requests over your interface endpoint. This approach allows you to optimize your networking costs by using gateway endpoints, which are not billed, for your in-VPC traffic. Your on-premises applications can use AWS PrivateLink with the help of the inbound Resolver endpoint. Amazon provides a DNS server, called the Route 53 Resolver, for your VPC. An inbound Resolver endpoint forwards DNS queries from the on-premises network to Route 53 Resolver.

Important

To take advantage of the lowest cost network path when using **Enable private DNS only for inbound endpoints**, a gateway endpoint must be present in your VPC. The presence of a gateway endpoint helps ensure that in-VPC traffic always routes over the AWS private network when the **Enable private DNS only for inbound endpoints** option is selected. You must maintain this gateway endpoint while you have the **Enable private DNS only for inbound endpoints** option selected. If you want to delete your gateway endpoint you must first clear **Enable private DNS only for inbound endpoints**.
If you want to update an existing interface endpoint to **Enable private DNS only for inbound endpoints**, first confirm that your VPC has an S3 gateway endpoint. For more information about gateway endpoints and managing private DNS names, see Gateway VPC endpoints and Manage DNS names respectively in the *AWS PrivateLink Guide*.
When enabling **Private DNS for inbound resolver only**, the `dnsRecordIpType` of the gateway endpoint must either match that of interface endpoint or be **service defined**.

The **Enable private DNS only for inbound endpoints** option is available only for services that support gateway endpoints.

For more information about creating a VPC endpoint that uses **Enable private DNS only for inbound endpoints**, see Create an interface endpoint in the *AWS PrivateLink Guide*.

Using the VPC console

In the console you have two options: **Enable DNS name** and **Enable private DNS only for inbound endpoints**. **Enable DNS name** is an option supported by AWS PrivateLink. By using the **Enable DNS name** option, you can use Amazon's private connectivity to Amazon S3, while making requests to the default public endpoint DNS names. When this option is enabled, customers can take advantage of the lowest cost network path available to their application.

When you enable private DNS names on an existing or new VPC interface endpoint for Amazon S3, the **Enable private DNS only for inbound endpoints** option is selected by default. If this option is selected, your applications use only interface endpoints for your on-premises traffic. This in-VPC traffic automatically uses the lower-cost gateway endpoints. Alternatively, you can clear **Enable private DNS only for inbound endpoints** to route all S3 requests over your interface endpoint.

Using the AWS CLI

If you don't specify a value for `PrivateDnsOnlyForInboundResolverEndpoint`, it will default to `true`. However, before your VPC applies your settings, it performs a check to make sure that you have a gateway endpoint present in the VPC. If a gateway endpoint is present in the VPC, the call succeeds. If not, you will see the following error message:

To set PrivateDnsOnlyForInboundResolverEndpoint to true, the VPC *vpce_id* must have a gateway endpoint for the service.

For a new VPC Interface endpoint

Use the `private-dns-enabled` and `dns-options` attributes to enable private DNS through the command line. The `PrivateDnsOnlyForInboundResolverEndpoint` option in the `dns-options` attribute must be set to `true`. Replace the *user input placeholders* with your own information.

```
aws ec2 create-vpc-endpoint \
--region us-east-1 \
--service-name s3-service-name \
--vpc-id client-vpc-id \
--subnet-ids client-subnet-id \
--vpc-endpoint-type Interface  \
--private-dns-enabled  \
--ip-address-type ip-address-type \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true \
--security-group-ids client-sg-id
```

For an existing VPC endpoint

If you want to use private DNS for an existing VPC endpoint, use the following example command and replace the *user input placeholders* with your own information.

```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=false
```

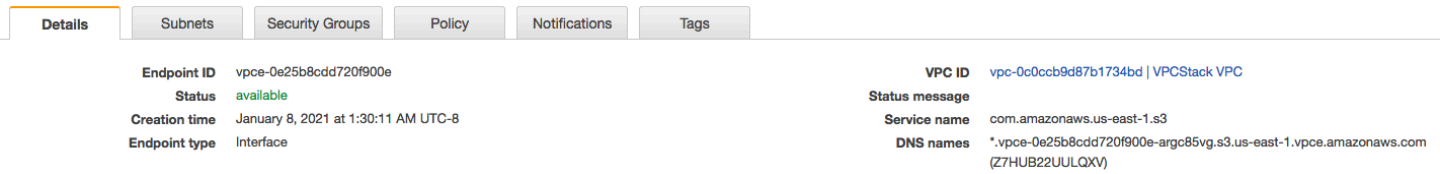
If you want to update an existing VPC endpoint to enable private DNS only for the Inbound Resolver, use the following example and replace the sample values with your own.

```
aws ec2 modify-vpc-endpoint \
--region us-east-1 \
--vpc-endpoint-id client-vpc-id \
--private-dns-enabled \
--dns-options PrivateDnsOnlyForInboundResolverEndpoint=true
```

Accessing buckets, access points, and Amazon S3 Control API operations from S3 interface endpoints

You can use the AWS CLI or AWS SDKs to access buckets, S3 access points, and Amazon S3 Control API operations through S3 interface endpoints.

The following image shows the VPC console **Details** tab, where you can find the DNS name of a VPC endpoint. In this example, the *VPC endpoint ID (vpce-id)* is `vpce-0e25b8cdd720f900e` and the *DNS name* is `*.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com`.



When using the DNS name to access a resource, replace `*` with the appropriate value. The appropriate values to use in place of `*` are as follows:

- `bucket`
- `accesspoint`
- `control`

For example, to access a bucket, use a *DNS name* like this:

`bucket.vpce-0e25b8cdd720f900e-argc85vg.s3.us-east-1.vpce.amazonaws.com`

For examples of how to use DNS names to access buckets, access points, and Amazon S3 Control API operations, see the following sections of AWS CLI examples and AWS SDK examples.

For more information about how to view your endpoint-specific DNS names, see Viewing endpoint service private DNS name configuration in the *VPC User Guide*.

AWS CLI examples

To access S3 buckets, S3 access points, or Amazon S3 Control API operations through S3 interface endpoints in AWS CLI commands, use the `--region` and `--endpoint-url` parameters.

Example: Use an endpoint URL to list objects in your bucket

In the following example, replace the bucket name *my-bucket*, Region *us-east-1*, and the DNS name of the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
aws s3 ls s3://my-bucket/ --region us-east-1 --endpoint-url
 https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
```

Example: Use an endpoint URL to list objects from an access point

- **Method 1** – Using the Amazon Resource Name (ARN) of the access point with the access point endpoint

  Replace the ARN *us-east-1:123456789012:accesspoint/accesspointexamplename*, the Region *us-east-1*, and the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

  ```
  aws s3api list-objects-v2 --bucket arn:aws:s3:us-east-1:123456789012:accesspoint/
  accesspointexamplename --region us-east-1 --endpoint-url
   https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com
  ```

  If you can't run the command successfully, update your AWS CLI to the latest version and try again. For more information on the update instructions, see Installing or updating the latest version of the AWS CLI in the *AWS Command Line Interface User Guide*.
- **Method 2** – Using the alias of the access point with the regional bucket endpoint

  In the following example, replace the access point alias *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*,

the Region *us-east-1*, and the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
aws s3api list-objects-v2 --
bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias
 --region us-east-1 --endpoint-url https://bucket.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com
```

- **Method 3** – Using the alias of the access point with the access point endpoint

  First, to construct an S3 endpoint with the bucket included as part of the hostname, set the addressing style to `virtual` for `aws s3api` to use. For more information about `AWS configure`, see Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

  ```
  aws configure set default.s3.addressing_style virtual
  ```

  Then, in the following example, replace the access point alias *accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias*, the Region *us-east-1*, and the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information. For more information about access point alias, see Access point aliases.

  ```
  aws s3api list-objects-v2 --
  bucket accesspointexamplename-8tyekmigicmhun8n9kwpfur39dnw4use1a-s3alias --
  region us-east-1 --endpoint-url https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
  east-1.vpce.amazonaws.com
  ```

Example: Use an endpoint URL to list jobs with an S3 control API operation

In the following example, replace the Region *us-east-1*, the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com*, and the account ID *12345678* with your own information.

```
aws s3control --region us-east-1 --endpoint-url
 https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com list-jobs --
account-id 12345678
```

AWS SDK examples

To access S3 buckets, S3 access points, or Amazon S3 Control API operations through S3 interface endpoints when using the AWS SDKs, update your SDKs to the latest version. Then configure your clients to use an endpoint URL for accessing a bucket, access point, or Amazon S3 Control API operations through S3 interface endpoints.

SDK for Python (Boto3)

Example: Use an endpoint URL to access an S3 bucket

In the following example, replace the Region *us-east-1* and VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
s3_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
)
```

Example: Use an endpoint URL to access an S3 access point

In the following example, replace the Region *us-east-1* and VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
ap_client = session.client(
service_name='s3',
region_name='us-east-1',
endpoint_url='https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com'
)
```

Example: Use an endpoint URL to access the Amazon S3 Control API

In the following example, replace the Region *us-east-1* and VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
control_client = session.client(
service_name='s3control',
region_name='us-east-1',
endpoint_url='https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com'
```

```
)
```

SDK for Java 1.x

Example: Use an endpoint URL to access an S3 bucket

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
// bucket client
final AmazonS3 s3 = AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://bucket.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
List<Bucket> buckets = s3.listBuckets();
```

Example: Use an endpoint URL to access an S3 access point

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* and ARN *us-east-1:123456789012:accesspoint/prod* with your own information.

```
// accesspoint client
final AmazonS3 s3accesspoint =
 AmazonS3ClientBuilder.standard().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
ObjectListing objects = s3accesspoint.listObjects("arn:aws:s3:us-
east-1:123456789012:accesspoint/prod");
```

Example: Use an endpoint URL to access an Amazon S3 Control API operation

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* with your own information.

```
// control client
```

```
final AWSS3Control s3control =
 AWSS3ControlClient.builder().withEndpointConfiguration(
        new AwsClientBuilder.EndpointConfiguration(
                "https://control.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com",
                Regions.DEFAULT_REGION.getName()
        )
).build();
final ListJobsResult jobs = s3control.listJobs(new ListJobsRequest());
```

SDK for Java 2.x

Example: Use an endpoint URL to access an S3 bucket

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* and the Region *Region.US_EAST_1* with your own information.

```
// bucket client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://bucket.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com"))
                   .build()
```

Example: Use an endpoint URL to access an S3 access point

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* and the Region *Region.US_EAST_1* with your own information.

```
// accesspoint client
Region region = Region.US_EAST_1;
s3Client = S3Client.builder().region(region)

 .endpointOverride(URI.create("https://accesspoint.vpce-1a2b3c4d-5e6f.s3.us-
east-1.vpce.amazonaws.com"))
                   .build()
```

Example: Use an endpoint URL to access the Amazon S3 Control API

In the following example, replace the VPC endpoint ID *vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com* and the Region *Region.US_EAST_1* with your own information.

```
// control client
Region region = Region.US_EAST_1;
s3ControlClient = S3ControlClient.builder().region(region)

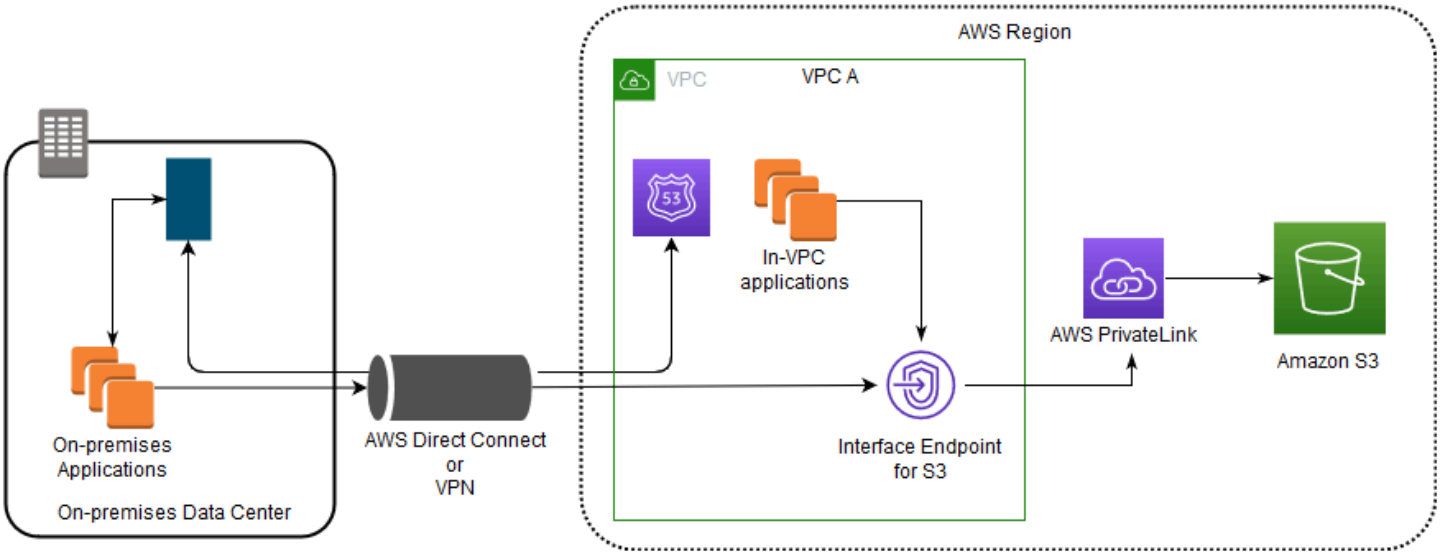
 .endpointOverride(URI.create("https://control.vpce-1a2b3c4d-5e6f.s3.us-east-1.vpce.amazonaws.com"))
                                  .build()
```

Updating an on-premises DNS configuration

When using endpoint-specific DNS names to access the interface endpoints for Amazon S3, you don't have to update your on-premises DNS resolver. You can resolve the endpoint-specific DNS name with the private IP address of the interface endpoint from the public Amazon S3 DNS domain.

Using interface endpoints to access Amazon S3 without a gateway endpoint or an internet gateway in the VPC

Interface endpoints in your VPC can route both in-VPC applications and on-premises applications to Amazon S3 over the Amazon network, as illustrated in the following diagram.

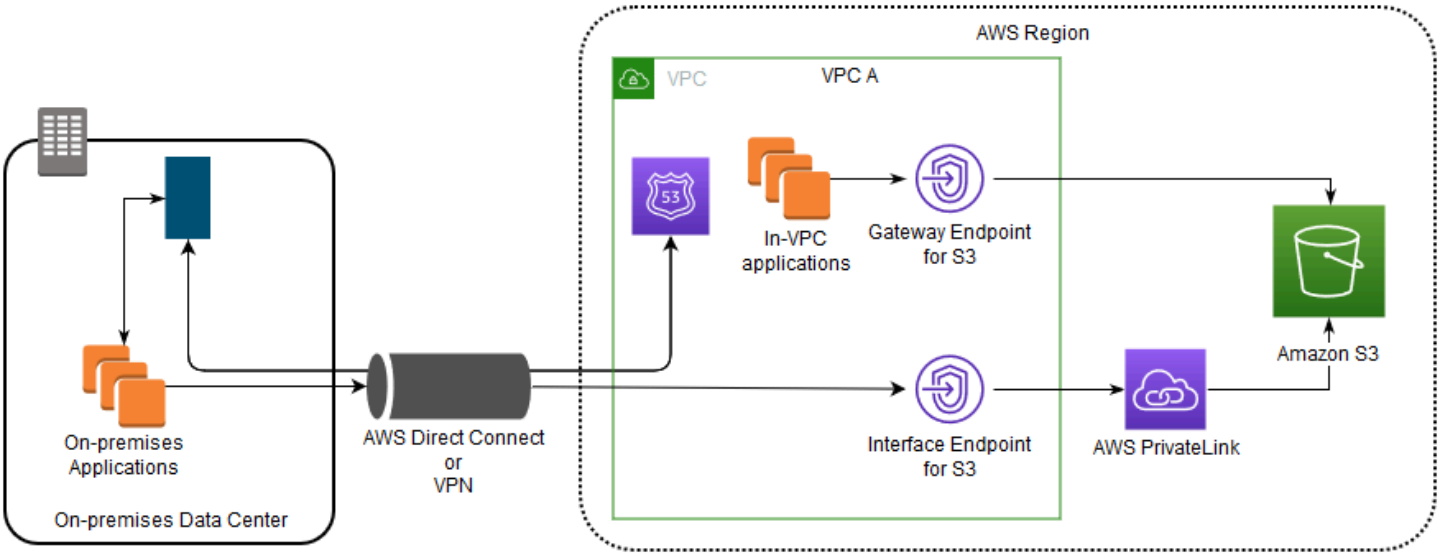


The diagram illustrates the following:

- Your on-premises network uses Direct Connect or Site-to-Site VPN to connect to VPC A.
- Your applications on-premises and in VPC A use endpoint-specific DNS names to access Amazon S3 through the S3 interface endpoint.
- On-premises applications send data to the interface endpoint in the VPC through Direct Connect (or Site-to-Site VPN). AWS PrivateLink moves the data from the interface endpoint to Amazon S3 over the AWS network.
- In-VPC applications also send traffic to the interface endpoint. AWS PrivateLink moves the data from the interface endpoint to Amazon S3 over the AWS network.

Using gateway endpoints and interface endpoints together in the same VPC to access Amazon S3

You can create interface endpoints and retain the existing gateway endpoint in the same VPC, as the following diagram shows. By taking this approach, you allow in-VPC applications to continue accessing Amazon S3 through the gateway endpoint, which is not billed. Then, only your on-premises applications would use interface endpoints to access Amazon S3. To access Amazon S3 this way, you must update your on-premises applications to use endpoint-specific DNS names for Amazon S3.



The diagram illustrates the following:

- On-premises applications use endpoint-specific DNS names to send data to the interface endpoint within the VPC through Direct Connect (or Site-to-Site VPN). AWS PrivateLink moves the data from the interface endpoint to Amazon S3 over the AWS network.

- Using default Regional Amazon S3 names, in-VPC applications send data to the gateway endpoint that connects to Amazon S3 over the AWS network.

For more information about gateway endpoints, see Gateway VPC endpoints in the *VPC User Guide*.

Creating a VPC endpoint policy for Amazon S3

You can attach an endpoint policy to your VPC endpoint that controls access to Amazon S3. The policy specifies the following information:

- The AWS Identity and Access Management (IAM) principal that can perform actions
- The actions that can be performed
- The resources on which actions can be performed

You can also use Amazon S3 bucket policies to restrict access to specific buckets from a specific VPC endpoint by using the `aws:sourceVpce` condition in your bucket policy. The following examples show policies that restrict access to a bucket or to an endpoint.

Topics

- Example: Restricting access to a specific bucket from a VPC endpoint
- Example: Restricting access to buckets in a specific account from a VPC endpoint
- Example: Restricting access to a specific VPC endpoint in the S3 bucket policy

Example: Restricting access to a specific bucket from a VPC endpoint

You can create an endpoint policy that restricts access to only specific Amazon S3 buckets. This type of policy is useful if you have other AWS services in your VPC that use buckets. The following bucket policy restricts access to only the *`amzn-s3-demo-bucket1`*. To use this endpoint policy, replace *`amzn-s3-demo-bucket1`* with the name of your bucket.

JSON

```
{
  "Version":"2012-10-17",
  "Id": "Policy1415115909151",
  "Statement": [
    { "Sid": "Access-to-specific-bucket-only",
```

```
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket1",
                   "arn:aws:s3:::amzn-s3-demo-bucket1/*"]
    }
  ]
}
```

Example: Restricting access to buckets in a specific account from a VPC endpoint

You can create an endpoint policy that restricts access to only the S3 buckets in a specific AWS account. To prevent clients within your VPC from accessing buckets that you don't own, use the following statement in your endpoint policy. The following example statement creates a policy that restricts access to resources owned by a single AWS account ID, *111122223333*.

```
{
  "Statement": [
    {
      "Sid": "Access-to-bucket-in-specific-account-only",
      "Principal": "*",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Deny",
      "Resource": "arn:aws:s3:::*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": "111122223333"
        }
      }
    }
  ]
}
```

Note

To specify the AWS account ID of the resource being accessed, you can use either the `aws:ResourceAccount` or the `s3:ResourceAccount` key in your IAM policy. However, be aware that some AWS services rely on access to AWS managed buckets. Therefore, using the `aws:ResourceAccount` or `s3:ResourceAccount` key in your IAM policy might also affect access to these resources.

Example: Restricting access to a specific VPC endpoint in the S3 bucket policy

The following Amazon S3 bucket policy allows access to a specific bucket, *amzn-s3-demo-bucket2*, from only the VPC endpoint *vpce-1a2b3c4d*. The policy denies all access to the bucket if the specified endpoint is not being used. The `aws:sourceVpce` condition specifies the endpoint and doesn't require an Amazon Resource Name (ARN) for the VPC endpoint resource, only the endpoint ID. To use this bucket policy, replace *amzn-s3-demo-bucket2* and *vpce-1a2b3c4d* with your bucket name and endpoint.

Important

- When applying the following Amazon S3 bucket policy to restrict access to only certain VPC endpoints, you might block your access to the bucket without intending to do so. Bucket policies that are intended to specifically limit bucket access to connections originating from your VPC endpoint can block all connections to the bucket. For information about how to fix this issue, see My bucket policy has the wrong VPC or VPC endpoint ID. How can I fix the policy so that I can access the bucket? in the *Support Knowledge Center*.
- Before using the following example policy, replace the VPC endpoint ID with an appropriate value for your use case. Otherwise, you won't be able to access your bucket.
- This policy disables *console* access to the specified bucket, because console requests don't originate from the specified VPC endpoint.

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Id": "Policy1415115909152",
  "Statement": [
    { "Sid": "Access-to-specific-VPCE-only",
      "Principal": "*",
      "Action": "s3:*",
      "Effect": "Deny",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket2",
                   "arn:aws:s3:::amzn-s3-demo-bucket2/*"],
      "Condition": {"StringNotEquals": {"aws:sourceVpce": "vpce-1a2b3c4d"}}
    }
  ]
}
```

For more policy examples, see Endpoints for Amazon S3 in the *VPC User Guide*.

For more information about VPC connectivity, see Network-to-VPC connectivity options in the AWS whitepaper Amazon Virtual Private Cloud Connectivity Options.

Compliance validation for Amazon S3

The security and compliance of Amazon S3 is assessed by third-party auditors as part of multiple AWS compliance programs, including the following:

- System and Organization Controls (SOC)
- Payment Card Industry Data Security Standard (PCI DSS)
- Federal Risk and Authorization Management Program (FedRAMP)
- Health Insurance Portability and Accountability Act (HIPAA)

AWS provides a frequently updated list of AWS services in scope of specific compliance programs at AWS Services in Scope by Compliance Program.

Third-party audit reports are available for you to download using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

For more information about AWS compliance programs, see AWS Compliance Programs.

Your compliance responsibility when using Amazon S3 is determined by the sensitivity of your data, your organization's compliance objectives, and applicable laws and regulations. If your use of Amazon S3 is subject to compliance with standards like HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- Security and Compliance Quick Start Guides that discuss architectural considerations and steps for deploying security- and compliance-focused baseline environments on AWS.
- Architecting for HIPAA Security and Compliance outlines how companies use AWS to help them meet HIPAA requirements.
- AWS Compliance Resources provide several different workbooks and guides that might apply to your industry and location.
- AWS Config can be used to assess how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub provides you with a comprehensive view of your security state within AWS and helps you check your compliance with security industry standards and best practices.
- Locking objects with Object Lock can help you meet technical requirements of financial services regulators (such as the SEC, FINRA, and CFTC) that require write once, read many (WORM) data storage for certain types of books and records information.

- Cataloging and analyzing your data with S3 Inventory can help you audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs.

Resilience in Amazon S3

The AWS global infrastructure is built around Regions and Availability Zones. AWS Regions provide multiple, physically separated and isolated Availability Zones that are connected with low latency, high throughput, and highly redundant networking. These Availability Zones offer you an effective way to design and operate applications and databases. They are more highly available, fault tolerant, and scalable than traditional single data center infrastructures or multi-data center infrastructures. If you specifically need to replicate your data over greater geographic distances, you can use Replicating objects within and across Regions, which enables automatic, asynchronous copying of objects across buckets in different AWS Regions.

Each AWS Region has multiple Availability Zones. You can deploy your applications across multiple Availability Zones in the same Region for fault tolerance and low latency. Availability Zones are connected to each other with fast, private fiber-optic networking, enabling you to easily architect applications that automatically fail over between Availability Zones without interruption.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, Amazon S3 offers several features to help support your data resiliency and backup needs.

Lifecycle configuration

A lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. With lifecycle configuration rules, you can tell Amazon S3 to transition objects to less expensive storage classes, archive them, or delete them. For more information, see Managing the lifecycle of objects.

Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. For more information, see Retaining multiple versions of objects with S3 Versioning.

S3 Object Lock

You can use S3 Object Lock to store objects using a *write once, read many* (WORM) model. Using S3 Object Lock, you can prevent an object from being deleted or overwritten for a fixed amount of time or indefinitely. S3 Object Lock enables you to meet regulatory requirements

that require WORM storage or simply to add an additional layer of protection against object changes and deletion. For more information, see Locking objects with Object Lock.

Storage classes

Amazon S3 offers a range of storage classes to choose from depending on the requirements of your workload. The S3 Standard-IA and S3 One Zone-IA storage classes are designed for data you access about once a month and need milliseconds access. The S3 Glacier Instant Retrieval storage class is designed for long-lived archive data accessed with milliseconds access that you access about once a quarter. For archive data that does not require immediate access, such as backups, you can use the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. For more information, see Understanding and managing Amazon S3 storage classes.

The following security best practices also address resilience:

- Enable versioning
- Consider Amazon S3 cross-region replication
- Identify and audit all your Amazon S3 buckets

Encryption of Amazon S3 backups

If you are storing backups using Amazon S3, the encryption of your backups depends on the configuration of those buckets. Amazon S3 provides a way to set the default encryption behavior for an S3 bucket. You can set default encryption on a bucket so that all objects are encrypted when they are stored in the bucket. The default encryption supports keys stored in AWS KMS (SSE-KMS). For more information, see Setting default server-side encryption behavior for Amazon S3 buckets.

For more information about Versioning and Object Lock, see the following topics: Retaining multiple versions of objects with S3 Versioning Locking objects with Object Lock

Infrastructure security in Amazon S3

As a managed service, Amazon S3 is protected by the AWS global network security procedures that are described in the security pillar of the AWS Well-Architected Framework.

Access to Amazon S3 through the network is through AWS published APIs. Clients must support Transport Layer Security (TLS) 1.2. We recommend also supporting TLS 1.3 and hybrid post-quantum key exchange. To learn about post-quantum cryptography at AWS, including links to blog posts and research papers, see Post-Quantum Cryptography for AWS.

> **Note**
>
> TLS 1.3 is supported in all S3 endpoints, except for AWS PrivateLink for Amazon S3 and Multi-Region Access Points.

Clients must also support cipher suites with Perfect Forward Secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). Additionally, requests must be signed using AWS Signature V4 or AWS Signature V2, requiring valid credentials to be provided.

These APIs are callable from any network location. However, Amazon S3 does support resource-based access policies, which can include restrictions based on the source IP address. You can also use Amazon S3 bucket policies to control access to buckets from specific virtual private cloud (VPC) endpoints, or specific VPCs. Effectively, this isolates network access to a given Amazon S3 bucket from only the specific VPC within the AWS network. For more information, see Controlling access from VPC endpoints with bucket policies.

The following security best practices also address infrastructure security in Amazon S3:

- Consider VPC endpoints for Amazon S3 access
- Identify and audit all your Amazon S3 buckets

Configuration and vulnerability analysis in Amazon S3

AWS handles basic security tasks like guest operating system (OS) and database patching, firewall configuration, and disaster recovery. These procedures have been reviewed and certified by the appropriate third parties. For more details, see the following resources:

- Compliance validation for Amazon S3
- Shared Responsibility Model
- Amazon Web Services: Overview of Security Processes

The following security best practices also address configuration and vulnerability analysis in Amazon S3:

- Identify and audit all your Amazon S3 buckets
- Enable AWS Config

Access management

Amazon S3 provides a variety of access management tools. The following is a list of these features and tools. You do not need all of these access management tools, but you must use one or more to grant access to your Amazon S3 buckets, objects, and other S3 resources. Proper application of these tools can help make sure that your resources are accessible only to the intended users.

The most commonly used access management tool is an *access policy*. An access policy can be a *resource-based policy* that is attached to an AWS resource, such as a bucket policy for a bucket. An access policy can also be an *identity-based policy* that is attached to an AWS Identity and Access Management (IAM) identity, such as an IAM user, group, or role. An access policy describes who has access to what things. Write an access policy to grant AWS accounts and IAM users, groups, and roles permission to perform operations on a resource. For example, you can grant `PUT Object` permission to another AWS account so that the other account can upload objects to your bucket.

The following are the access management tools available in Amazon S3. For a more comprehensive guide on Amazon S3 access control, see Access control in Amazon S3.

Bucket policy

An Amazon S3 bucket policy is a JSON-formatted AWS Identity and Access Management (IAM) resource-based policy that is attached to a particular bucket. Use bucket policies to grant other

AWS accounts or IAM identities permissions for the bucket and the objects in it. Many S3 access management use cases can be met by using a bucket policy. With bucket policies, you can personalize bucket access to help make sure that only the identities that you have approved can access resources and perform actions within them. For more information, see Bucket policies for Amazon S3.

Identity-based policy

An identity-based or IAM user policy is a type of AWS Identity and Access Management (IAM) policy. An identity-based policy is a JSON-formatted policy that is attached to IAM users, groups, or roles in your AWS account. You can use identity-based policies to grant an IAM identity access to your buckets or objects. You can create IAM users, groups, and roles in your account and attach access policies to them. You can then grant access to AWS resources, including Amazon S3 resources. For more information, see Identity-based policies for Amazon S3.

S3 Access Grants

Use S3 Access Grants to create access grants to your Amazon S3 data for both identities in corporate identity directories, such as Active Directory, and to AWS Identity and Access Management (IAM) identities. S3 Access Grants helps you manage data permissions at scale. Additionally, S3 Access Grants logs end-user identity and the application used to access the S3 data in AWS CloudTrail. This provides a detailed audit history down to the end-user identity for all access to the data in your S3 buckets. For more information, see Managing access with S3 Access Grants.

Access Points

Amazon S3 Access Points simplifies managing data access at scale for applications that use shared datasets on S3. Access Points are named network endpoints that are attached to a bucket. You can use access points to perform S3 object operations at scale, such as uploading and retrieving objects. A bucket can have up to 10,000 access points attached, and for each access point, you can enforce distinct permissions and network controls to give you detailed control over access to your S3 objects. S3 Access Points can be associated with buckets in the same account or in another trusted account. Access Points policies are resource-based policies that are evaluated in conjunction with the underlying bucket policy. For more information, see Managing access to shared datasets with access points.

Access control list (ACL)

An ACL is a list of grants identifying the grantee and the permission granted. ACLs grant basic read or write permissions to other AWS accounts. ACLs use an Amazon S3–specific XML schema. An ACL is a type of AWS Identity and Access Management (IAM) policy. An object ACL is used to manage access to an object, and a bucket ACL is used to manage access to a bucket. With bucket policies, there is a single policy for the entire bucket, but object ACLs are specified for each object. We recommend that you keep ACLs turned off, except in circumstances where you must individually control access for each object. For more information about using ACLs, see Controlling ownership of objects and disabling ACLs for your bucket.

> **Warning**
>
> The majority of modern use cases in Amazon S3 do not require the use of ACLs.

Object Ownership

To manage access to your objects, you must be the owner of the object. You can use the Object Ownership bucket-level setting to control ownership of objects uploaded to your bucket. Also, use Object Ownership to turn on ACLs. By default, Object Ownership is set to the *Bucket owner enforced setting* and all ACLs are turned off. When ACLs are turned off, the bucket owner owns all of the objects in the bucket and exclusively manages access to data. To manage access, the bucket owner uses policies or another access management tool, excluding ACLs. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

For a more comprehensive guide on Amazon S3 access control and additional best practices, see Access control in Amazon S3.

Amazon Simple Storage Service data inventory

Amazon S3

Amazon S3 provides scalable object storage in the AWS Cloud. It allows you to store and retrieve any amount of data from anywhere on the web. Based on its unique architecture, S3 is designed to exceed 99.999999999% (11 nines) data durability. Additionally, S3 stores data redundantly across a minimum of 3 Availability Zones by default, providing built-in resilience against widespread disaster. Customers can store data in a single Availability Zone to minimize storage cost or latency, in multiple Availability Zones for resilience against the permanent loss of an entire data center, or in multiple AWS Regions to meet geographic resilience requirements.

Key Characteristics

Geographic location

Amazon S3 is hosted in multiple locations worldwide. You select locations for your data that put them close to your customers.

Buckets

A bucket is a container for objects stored in Amazon S3. Every object is contained within a bucket.

Objects

Objects are the fundamental entities stored in Amazon S3. Objects consist of object data and metadata.

Storage Classes

Amazon S3 offers different storage classes optimized for different use cases.

Storage Management

Amazon S3 has storage management features that you can use to manage costs and adhere to compliance requirements.

Access management and security

Amazon S3 provides features for auditing and managing access to your buckets and objects.

Geographic location

Amazon S3 is available in every AWS Region worldwide. Each Region is a separate geographic area.

Why this matters

After you determine where you want to store your data, you can decide whether to deploy functionally equivalent storage in the same locations or different locations, depending on your needs.

To get a summary of your Amazon S3 buckets across all Regions

Use the following AWS CLI command:

```
aws s3api list-buckets /
 --max-items 100 /
 --page-size 100
```

For more information, see list-buckets in the AWS CLI Command Reference.

Buckets

Amazon S3 buckets are containers for objects. Each bucket has a unique name across all of AWS. Amazon S3 supports four types of buckets: general purpose buckets, Directory buckets, table buckets, and vector buckets. Each type of bucket provides a unique set of features for different use cases. For more information on the different bucket types, see Buckets in the Amazon S3 User Guide.

Why these matter

After you list your buckets, you can validate your buckets settings for functionally equivalent storage systems by reviewing the various bucket configuration settings.

To list bucket configurations

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket1
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket1
aws s3api get-bucket-logging --bucket amzn-s3-demo-bucket1
```

For more information, see get-bucket-versioning in the AWS CLI Command Reference.

Objects

Objects are the fundamental entities stored in Amazon S3. Each object consists of data, a key (name), and metadata.

Why these matter

Understanding your object characteristics helps plan for equivalent storage capacity and performance requirements in functionally equivalent systems.

To list objects and their properties

```
aws s3api list-objects-v2 --bucket amzn-s3-demo-bucket1 /
```

```
    --query 'Contents[].{Key: Key, Size: Size, LastModified: LastModified}'
```

For more information, see list-objects-v2 in the AWS CLI Command Reference.

Storage Classes

- Amazon S3 Standard
- Amazon S3 Intelligent-Tiering
- Amazon S3 Standard-IA
- Amazon S3 One Zone-IA
- Amazon S3 Glacier Instant Retrieval
- Amazon S3 Glacier Flexible Retrieval
- Amazon S3 Glacier Deep Archive
- Amazon S3 Express One Zone

Why these matter

Understanding your storage class usage helps determine appropriate storage tiers in functionally equivalent systems. For more information, see Understanding and managing Amazon S3 storage classes in the Amazon S3 User Guide.

To review storage class selection and usage

You can use Amazon S3 Storage Lens to review your storage class selection and usage. For more information, see Understanding and manage Amazon S3 storage classes in the Amazon S3 User Guide.

Storage management

Why this matters

Understanding your storage management usage helps plan for equivalent features to manage costs and adhere to compliance requirements in functionally equivalent systems.

To review storage managment feature selection and usage

You can use Amazon S3 Storage Lens to review your usage of storage management features. For more information, see Amazon S3 Storage Lens metrics glossary in the Amazon S3 User Guide.

Access management and security

Why these matter

Understanding your access management and security settings help you plan for equivalent features to manage access and security requirements in functionally equivalent systems.

To review your access management and security settings

After you list your buckets, you can validate your buckets security and access settings for functionally equivalent storage systems by reviewing the various bucket configuration settings.

```
aws s3api get-public-access-block --bucket amzn-s3-demo-bucket1
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket1
aws s3api get-bucket-encryption --bucket amzn-s3-demo-bucket1
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket1
```

Data Transfer

You can transfer data from Amazon S3 using several methods:

- AWS CLI
- AWS SDK
- Amazon S3 REST API
- Third-party tools

Example using AWS CLI

To download an entire bucket:

```
aws s3 sync s3://amzn-s3-demo-bucket1 /local/path
```

For more information, see sync in the AWS CLI Command Reference.

To download specific objects:

```
aws s3 cp s3://amzn-s3-demo-bucket1/path/to/object /local/path
```

For more information, see cp in the AWS CLI Command Reference.

Related Resources

The following are additional characteristics of Amazon S3:

- Access control in Amazon S3
- Security in Amazon S3
- Data protection in Amazon S3
- Logging and monitoring in Amazon S3

Data protection in Amazon S3

In addition to the resilience offered by the AWS global infrastructure, Amazon S3 offers a number of features to help protect your data against accidental deletions or Regional failures.

S3 Replication

You can use live replication to enable automatic, asynchronous copying of objects across Amazon S3 buckets. Buckets that are configured for object replication can be owned by the same AWS account or by different accounts. You can replicate objects to a single destination bucket or to multiple destination buckets. The destination buckets can be in different AWS Regions or within the same Region as the source bucket. To enable failover controls, you can configure replication to be two-way (bidirectional) so that your source and destination buckets can be kept in sync during a Regional failure. For more information, see the section called “Replicating objects within and across Regions”.

Multi-Region Access Points and failover controls

Amazon S3 Multi-Region Access Points provide a global endpoint that applications can use to fulfill requests from S3 buckets that are located in multiple AWS Regions. You can use Multi-Region Access Points to build multi-Region applications with the same architecture that's used in a single Region, and then run those applications anywhere in the world. Instead of sending requests over the congested public internet, Multi-Region Access Points provide built-in network resilience with acceleration of internet-based requests to Amazon S3. Application requests made to a Multi-Region Access Point global endpoint use AWS Global Accelerator to automatically route over the AWS global network to the closest-proximity S3 bucket with an active routing status. For more information about Multi-Region Access Points, see the section called “Managing multi-region traffic ”.

With Amazon S3 Multi-Region Access Point failover controls, you can maintain business continuity during Regional traffic disruptions, while also giving your applications a multi-Region architecture to fulfill compliance and redundancy needs. If your Regional traffic gets disrupted, you can use Multi-Region Access Point failover controls to select which AWS Regions behind an Amazon S3 Multi-Region Access Point will process data-access and storage requests.

To support failover, you can set up your Multi-Region Access Point in an active-passive configuration, with traffic flowing to the active Region during normal conditions, and a passive Region on standby for failover. If you have S3 Cross-Region Replication (CRR) enabled with

two-way replication rules, you can keep your buckets synchronized during a failover. For more information about failover controls, see the section called “Failover configuration”.

S3 Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. For more information, see Retaining multiple versions of objects with S3 Versioning.

S3 Object Lock

You can use S3 Object Lock to store objects using a *write once, read many* (WORM) model. Using S3 Object Lock, you can prevent an object from being deleted or overwritten for a fixed amount of time or indefinitely. S3 Object Lock enables you to meet regulatory requirements that require WORM storage or simply to add an additional layer of protection against object changes and deletion. For more information, see Locking objects with Object Lock.

AWS Backup

Amazon S3 is natively integrated with AWS Backup, a fully managed, policy-based service that you can use to centrally define backup policies to protect your data in Amazon S3. After you define your backup policies and assign Amazon S3 resources to the policies, AWS Backup automates the creation of Amazon S3 backups and securely stores the backups in an encrypted backup vault that you designate in your backup plan. For more information, see the section called “Backing up your data”.

For a tutorial on using some of these features together to protect your data, see Tutorial: Protecting data on Amazon S3 against accidental deletion or application bugs using S3 Versioning, S3 Object Lock, and S3 Replication.

Important

In addition to using the preceding features to protect your data, we recommend reviewing the recommendations in the section called “Security best practices”.

Topics

- Replicating objects within and across Regions

- Managing multi-Region traffic with Multi-Region Access Points
- Retaining multiple versions of objects with S3 Versioning
- Locking objects with Object Lock
- Backing up your Amazon S3 data

Replicating objects within and across Regions

You can use replication to enable automatic, asynchronous copying of objects across Amazon S3 buckets. Buckets that are configured for object replication can be owned by the same AWS account or by different accounts. You can replicate objects to a single destination bucket or to multiple destination buckets. The destination buckets can be in different AWS Regions or within the same Region as the source bucket.

There are two types of replication: *live replication* and *on-demand replication*.

- **Live replication** – **To automatically replicate new and updated objects** as they are written to the source bucket, use live replication. Live replication doesn't replicate any objects that existed in the bucket before you set up replication. To replicate objects that existed before you set up replication, use on-demand replication.
- **On-demand replication** – **To replicate existing objects** from the source bucket to one or more destination buckets on demand, use S3 Batch Replication. For more information about replicating existing objects, see When to use S3 Batch Replication.

There are two forms of live replication: *Cross-Region Replication (CRR)* and *Same-Region Replication (SRR)*.

- **Cross-Region Replication (CRR)** – You can use CRR to replicate objects across Amazon S3 buckets in different AWS Regions. For more information about CRR, see the section called “When to use Cross-Region Replication”.
- **Same-Region Replication (SRR)** – You can use SRR to copy objects across Amazon S3 buckets in the same AWS Region. For more information about SRR, see the section called “When to use Same-Region Replication”.

Topics

- Why use replication?

- When to use Cross-Region Replication
- When to use Same-Region Replication
- When to use two-way replication (bi-directional replication)
- When to use S3 Batch Replication
- Workload requirements and live replication
- What does Amazon S3 replicate?
- Requirements and considerations for replication
- Setting up live replication overview
- Managing or pausing live replication
- Replicating existing objects with Batch Replication
- Troubleshooting replication
- Monitoring replication with metrics, event notifications, and statuses

Why use replication?

Replication can help you do the following:

- **Replicate objects while retaining metadata** – You can use replication to make copies of your objects that retain all metadata, such as the original object creation times and version IDs. This capability is important if you must ensure that your replica is identical to the source object.
- **Replicate objects into different storage classes** – You can use replication to directly put objects into S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, or another storage class in the destination buckets. You can also replicate your data to the same storage class and use lifecycle configurations on the destination buckets to move your objects to a colder storage class as they age.
- **Maintain object copies under different ownership** – Regardless of who owns the source object, you can tell Amazon S3 to change replica ownership to the AWS account that owns the destination bucket. This is referred to as the *owner override* option. You can use this option to restrict access to object replicas.
- **Keep objects stored over multiple AWS Regions** – To ensure geographic differences in where your data is kept, you can set multiple destination buckets across different AWS Regions. This feature might help you meet certain compliance requirements.
- **Replicate objects within 15 minutes** – To replicate your data in the same AWS Region or across different Regions within a predictable time frame, you can use S3 Replication Time Control (S3

RTC). S3 RTC replicates 99.99 percent of new objects stored in Amazon S3 within 15 minutes (backed by a service-level agreement). For more information, see the section called "Using S3 Replication Time Control".

> **Note**
>
> S3 RTC does not apply to Batch Replication. Batch Replication is an on-demand replication job, and can be tracked with S3 Batch Operations. For more information, see Tracking job status and completion reports.

- **Sync buckets, replicate existing objects, and replicate previously failed or replicated objects** – To sync buckets and replicate existing objects, use Batch Replication as an on-demand replication action. For more information about when to use Batch Replication, see When to use S3 Batch Replication.
- **Replicate objects and fail over to a bucket in another AWS Region** – To keep all metadata and objects in sync across buckets during data replication, use two-way replication (also known as bi-directional replication) rules before configuring Amazon S3 Multi-Region Access Point failover controls. Two-way replication rules help ensure that when data is written to the S3 bucket that traffic fails over to, that data is then replicated back to the source bucket.

When to use Cross-Region Replication

S3 Cross-Region Replication (CRR) is used to copy objects across Amazon S3 buckets in different AWS Regions. CRR can help you do the following:

- **Meet compliance requirements** – Although Amazon S3 stores your data across multiple geographically distant Availability Zones by default, compliance requirements might dictate that you store data at even greater distances. To satisfy these requirements, use Cross-Region Replication to replicate data between distant AWS Regions.
- **Minimize latency** – If your customers are in two geographic locations, you can minimize latency in accessing objects by maintaining object copies in AWS Regions that are geographically closer to your users.
- **Increase operational efficiency** – If you have compute clusters in two different AWS Regions that analyze the same set of objects, you might choose to maintain object copies in those Regions.

When to use Same-Region Replication

Same-Region Replication (SRR) is used to copy objects across Amazon S3 buckets in the same AWS Region. SRR can help you do the following:

- **Aggregate logs into a single bucket** – If you store logs in multiple buckets or across multiple accounts, you can easily replicate logs into a single, in-Region bucket. Doing so allows for simpler processing of logs in a single location.
- **Configure live replication between production and test accounts** – If you or your customers have production and test accounts that use the same data, you can replicate objects between those multiple accounts, while maintaining object metadata.
- **Abide by data sovereignty laws** – You might be required to store multiple copies of your data in separate AWS accounts within a certain Region. Same-Region Replication can help you automatically replicate critical data when compliance regulations don't allow the data to leave your country.

When to use two-way replication (bi-directional replication)

- **Build shared datasets across multiple AWS Regions** – With replica modification sync, you can easily replicate metadata changes, such as object access control lists (ACLs), object tags, or object locks, on replication objects. This two-way replication is important if you want to keep all objects and object metadata changes in sync. You can enable replica modification sync on a new or existing replication rule when performing two-way replication between two or more buckets in the same or different AWS Regions.
- **Keep data synchronized across Regions during failover** – You can synchronize data in buckets between AWS Regions by configuring two-way replication rules with S3 Cross-Region Replication (CRR) directly from a Multi-Region Access Point. To make an informed decision on when to initiate failover, you can also enable S3 replication metrics so that you can monitor the replication in Amazon CloudWatch, in S3 Replication Time Control (S3 RTC), or from the Multi-Region Access Point.
- **Make your application highly available** – Even in the event of a Regional traffic disruption, you can use two-way replication rules to keep all metadata and objects in sync across buckets during data replication.

When to use S3 Batch Replication

Batch Replication replicates existing objects to different buckets as an on-demand option. Unlike live replication, these jobs can be run as needed. Batch Replication can help you do the following:

- **Replicate existing objects** – You can use Batch Replication to replicate objects that were added to the bucket before Same-Region Replication or Cross-Region Replication were configured.
- **Replicate objects that previously failed to replicate** – You can filter a Batch Replication job to attempt to replicate objects with a replication status of **FAILED**.
- **Replicate objects that were already replicated** – You might be required to store multiple copies of your data in separate AWS accounts or AWS Regions. Batch Replication can replicate existing objects to newly added destinations.
- **Replicate replicas of objects that were created from a replication rule** – Replication configurations create replicas of objects in destination buckets. Replicas of objects can be replicated only with Batch Replication.

Workload requirements and live replication

Depending on your workload requirements, some types of live replication will be better suited to your use case than others. Use the following table to determine which type of replication to use for your situation, and whether to use S3 Replication Time Control (S3 RTC) for your workload. S3 RTC replicates 99.99 percent of new objects stored in Amazon S3 within 15 minutes (backed by a service-level agreement, or SLA). For more information, see the section called “Using S3 Replication Time Control”.

Workload requirement	S3 RTC (15-minute SLA)	Cross-Region Replication (CRR)	Same-Region Replication (SRR)
Replicate objects between different AWS accounts	Yes	Yes	Yes
Replicate objects within the same AWS Region within	No	No	Yes

Workload requirement	S3 RTC (15-minute SLA)	Cross-Region Replication (CRR)	Same-Region Replication (SRR)
24-48 hours (not SLA backed)			
Replicate objects between different AWS Regions within 24-48 hours (not SLA backed)	No	Yes	No
Predictable replication time: Backed by SLA to replicate 99.9 percent of objects within 15 minutes	Yes	No	No

What does Amazon S3 replicate?

Amazon S3 replicates only specific items in buckets that are configured for replication.

Topics

- What is replicated with replication configurations?
- What isn't replicated with replication configurations?

What is replicated with replication configurations?

By default, Amazon S3 replicates the following:

- Objects created after you add a replication configuration.
- Unencrypted objects.
- Objects encrypted using customer provided keys (SSE-C), objects encrypted at rest under an Amazon S3 managed key (SSE-S3) or a KMS key stored in AWS Key Management Service (SSE-KMS). For more information, see the section called “Replicating encrypted objects”.

- Object metadata from the source objects to the replicas. For information about replicating metadata from the replicas to the source objects, see Replicating metadata changes with replica modification sync.
- Only objects in the source bucket for which the bucket owner has permissions to read objects and access control lists (ACLs).

  For more information about resource ownership, see Amazon S3 bucket and object ownership.
- Object ACL updates, unless you direct Amazon S3 to change the replica ownership when source and destination buckets aren't owned by the same accounts.

  For more information, see Changing the replica owner.

  It can take a while until Amazon S3 can bring the two ACLs in sync. This change in ownership applies only to objects created after you add a replication configuration to the bucket.
- Object tags, if there are any.
- Object annotations, if there are any. Each annotation replicates independently as changes occur during live replication. During S3 Batch Replication, annotations are replicated independently alongside the object, but there are no separate job tasks for individual annotations. If annotation replication fails, the job task fails, but the object's replication status does not reflect the annotation failure.
- S3 Object Lock retention information, if there is any.

  When Amazon S3 replicates objects that have retention information applied, it applies those same retention controls to your replicas, overriding the default retention period configured on your destination buckets. If you don't have retention controls applied to the objects in your source bucket, and you replicate into destination buckets that have a default retention period set, the destination bucket's default retention period is applied to your object replicas. For more information, see Locking objects with Object Lock.

How delete operations affect replication

If you delete an object from the source bucket, the following actions occur by default:

- If you make a DELETE request without specifying an object version ID, Amazon S3 adds a delete marker. Amazon S3 deals with the delete marker as follows:
  - If you are using the latest version of the replication configuration (that is, you specify the `Filter` element in a replication configuration rule), Amazon S3 does not replicate the delete

marker by default. However, you can add *delete marker replication* to non-tag-based rules. For more information, see Replicating delete markers between buckets.

- If you don't specify the `Filter` element, Amazon S3 assumes that the replication configuration is version V1, and it replicates delete markers that resulted from user actions. However, if Amazon S3 deletes an object due to a lifecycle action, the delete marker is not replicated to the destination buckets.

- If you specify an object version ID to delete in a `DELETE` request, Amazon S3 deletes that object version in the source bucket. But it doesn't replicate the deletion in the destination buckets. In other words, it doesn't delete the same object version from the destination buckets. This protects data from malicious deletions.

What isn't replicated with replication configurations?

By default, Amazon S3 doesn't replicate the following:

- Objects in the source bucket that are replicas that were created by another replication rule. For example, suppose you configure replication where bucket A is the source and bucket B is the destination. Now suppose that you add another replication configuration where bucket B is the source and bucket C is the destination. In this case, objects in bucket B that are replicas of objects in bucket A are not replicated to bucket C.

  To replicate objects that are replicas, use Batch Replication. Learn more about configuring Batch Replication at Replicating existing objects.

- Objects in the source bucket that have already been replicated to a different destination. For example, if you change the destination bucket in an existing replication configuration, Amazon S3 won't replicate the objects again.

  To replicate previously replicated objects, use Batch Replication. Learn more about configuring Batch Replication at Replicating existing objects.

- Batch Replication does not support re-replicating objects that were deleted with the version ID of the object from the destination bucket. To re-replicate these objects, you can copy the source objects in place with a Batch Copy job. Copying those objects in place creates new versions of the objects in the source bucket and initiates replication automatically to the destination. For more information about how to use Batch Copy, see, Examples that use Batch Operations to copy objects.

- By default, when replicating from a different AWS account, delete markers added to the source bucket are not replicated.

  For information about how to replicate delete markers, see Replicating delete markers between buckets.

- Objects that are stored in the S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, S3 Intelligent-Tiering Archive Access, or S3 Intelligent-Tiering Deep Archive Access storage classes or tiers. You cannot replicate these objects until you restore them and copy them to a different storage class.

  To learn more about S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive, see Storage classes for rarely accessed objects.

  To learn more about the S3 Intelligent-Tiering, see Managing storage costs with Amazon S3 Intelligent-Tiering.

- Objects in the source bucket that the bucket owner doesn't have sufficient permissions to replicate.

  For information about how an object owner can grant permissions to a bucket owner, see Grant cross-account permissions to upload objects while ensuring that the bucket owner has full control.

- Updates to bucket-level subresources.

  For example, if you change the lifecycle configuration or add a notification configuration to your source bucket, these changes are not applied to the destination bucket. This feature makes it possible to have different configurations on source and destination buckets.

- Actions performed by lifecycle configuration.

  For example, if lifecycle configuration is enabled only on your source bucket, Amazon S3 creates delete markers for expired objects but doesn't replicate those markers. If you want the same lifecycle configuration applied to both the source and destination buckets, enable the same lifecycle configuration on both. For more information about lifecycle configuration, see Managing the lifecycle of objects.

- When you're using tag-based replication rules with live replication, new objects must be tagged with the matching replication rule tag in the `PutObject` operation. Otherwise, the objects won't be replicated. If objects are tagged after the `PutObject` operation, those objects also won't be replicated.

To replicate objects that have been tagged after the `PutObject` operation, you must use S3 Batch Replication. For more information about Batch Replication, see Replicating existing objects.

- Annotation deletions on the source bucket are not replicated to the destination. If you delete an annotation on the source, the annotation persists on the replica. You cannot exclude annotations from replication using replication rule filters. When replication is configured, all annotation creates and updates are replicated unless the replication role policy explicitly denies `s3:ReplicateObjectAnnotation`.

  If annotations are repeatedly deleted and recreated on the source with different names, the destination replica accumulates annotations over time because deletions are not replicated. This accumulation can eventually cause the destination to exceed the 1,000 annotation limit per object version.

  Annotation overwrites (updates to an existing annotation name) at the source are replicated to the destination. The destination annotation content stays current with the source.

Requirements and considerations for replication

Amazon S3 replication requires the following:

- The source bucket owner must have the source and destination AWS Regions enabled for their account. The destination bucket owner must have the destination Region enabled for their account.

  For more information about enabling or disabling an AWS Region, see Specify which AWS Regions your account can use in the *AWS Account Management Reference Guide*.
- Both source and destination buckets must have versioning enabled. For more information about versioning, see Retaining multiple versions of objects with S3 Versioning.
- Amazon S3 must have permissions to replicate objects from the source bucket to the destination bucket or buckets on your behalf. For more information about these permissions, see Setting up permissions for live replication.
- If the owner of the source bucket doesn't own the object in the bucket, the object owner must grant the bucket owner `READ` and `READ_ACP` permissions with the object access control list (ACL). For more information, see Access control list (ACL) overview.

- If the source bucket has S3 Object Lock enabled, the destination buckets must also have S3 Object Lock enabled.

  To enable replication on a bucket that has Object Lock enabled, you must use the AWS Command Line Interface, REST API, or AWS SDKs. For more general information, see Locking objects with Object Lock.

  > **Note**
  >
  > You must grant two new permissions on the source S3 bucket in the AWS Identity and Access Management (IAM) role that you use to set up replication. The two new permissions are `s3:GetObjectRetention` and `s3:GetObjectLegalHold`. If the role has an `s3:Get*` permission, it satisfies the requirement. For more information, see Setting up permissions for live replication.

For more information, see Setting up live replication overview.

If you are setting the replication configuration in a *cross-account scenario*, where the source and destination buckets are owned by different AWS accounts, the following additional requirement applies:

- The owner of the destination buckets must grant the owner of the source bucket permissions to replicate objects with a bucket policy. For more information, see (Optional) Step 3: Granting permissions when the source and destination buckets are owned by different AWS accounts.
- The destination buckets cannot be configured as Requester Pays buckets. For more information, see Using Requester Pays general purpose buckets for storage transfers and usage.

Considerations for replication

Before you create a replication configuration, be aware of the following considerations.

Topics

- Lifecycle configuration and object replicas
- Managing incomplete multipart uploads in destination buckets
- Versioning configuration and replication configuration
- Using S3 Replication with S3 Intelligent-Tiering

- Logging configuration and replication configuration
- CRR and the destination Region
- S3 Batch Replication
- S3 Replication Time Control

Lifecycle configuration and object replicas

The time it takes for Amazon S3 to replicate an object depends on the size of the object. For large objects, it can take several hours. Although it might take a while before a replica is available in the destination, it takes the same amount of time to create the replica as it took to create the corresponding object in the source bucket. If a lifecycle configuration is enabled on a destination bucket, the lifecycle rules honor the original creation time of the object, not when the replica became available in the destination bucket.

Replication configuration requires the bucket to be versioning-enabled. When you enable versioning on a bucket, keep the following in mind:

- If you have an object Expiration lifecycle configuration, after you enable versioning, add a `NonCurrentVersionExpiration` policy to maintain the same permanent delete behavior as before you enabled versioning.
- If you have a Transition lifecycle configuration, after you enable versioning, consider adding a `NonCurrentVersionTransition` policy.

You can also use an S3 Lifecycle rule to manage incomplete multipart uploads in destination buckets. For more information, see Managing incomplete multipart uploads in destination buckets.

Managing incomplete multipart uploads in destination buckets

When a source object was uploaded with multipart upload, Amazon S3 replicates it to the destination bucket with multipart upload. As with any multipart upload, if the upload isn't completed, its parts remain in the destination bucket until the upload is stopped. As a best practice, we recommend that you configure an S3 Lifecycle rule on the destination bucket by using the `AbortIncompleteMultipartUpload` action to minimize your storage costs. For more information, see Configuring a bucket lifecycle configuration to delete incomplete multipart uploads.

Versioning configuration and replication configuration

Both the source and destination buckets must be versioning-enabled when you configure replication on a bucket. After you enable versioning on both the source and destination buckets and configure replication on the source bucket, you will encounter the following issues:

- If you attempt to disable versioning on the source bucket, Amazon S3 returns an error. You must remove the replication configuration before you can disable versioning on the source bucket.
- If you disable versioning on the destination bucket, replication fails. The source object has the replication status `FAILED`.

Using S3 Replication with S3 Intelligent-Tiering

S3 Intelligent-Tiering is a storage class that is designed to optimize storage costs by automatically moving data to the most cost-effective access tier. For a small monthly object monitoring and automation charge, S3 Intelligent-Tiering monitors access patterns and automatically moves objects that have not been accessed to lower-cost access tiers.

Replicating objects stored in S3 Intelligent-Tiering with S3 Batch Replication or invoking CopyObject or UploadPartCopy constitutes access. In these cases, the source objects of the copy or replication operations are tiered up.

For more information about S3 Intelligent-Tiering see, Managing storage costs with Amazon S3 Intelligent-Tiering.

Logging configuration and replication configuration

If Amazon S3 delivers logs to a bucket that has replication enabled, it replicates the log objects.

If server access logs or AWS CloudTrail logs are enabled on your source or destination bucket, Amazon S3 includes replication-related requests in the logs. For example, Amazon S3 logs each object that it replicates.

CRR and the destination Region

Amazon S3 Cross-Region Replication (CRR) is used to copy objects across S3 buckets in different AWS Regions. You might choose the Region for your destination bucket based on either your business needs or cost considerations. For example, inter-Region data transfer charges vary depending on the Regions that you choose.

Suppose that you chose US East (N. Virginia) (`us-east-1`) as the Region for your source bucket. If you choose US West (Oregon) (`us-west-2`) as the Region for your destination buckets, you pay more than if you choose the US East (Ohio) (`us-east-2`) Region. For pricing information, see "Data Transfer Pricing" in Amazon S3 pricing.

There are no data transfer charges associated with Same-Region Replication (SRR).

S3 Batch Replication

For information about considerations for Batch Replication, see S3 Batch Replication considerations.

S3 Replication Time Control

For information about best practices and considerations for S3 Replication Time Control (S3 RTC), see Best practices and guidelines for S3 RTC.

Setting up live replication overview

> **Note**
>
> Objects that existed before you set up replication aren't replicated automatically. In other words, Amazon S3 doesn't replicate objects retroactively. To replicate objects that were created before your replication configuration, use S3 Batch Replication. For more information about configuring Batch Replication, see Replicating existing objects.

To enable live replication—Same-Region Replication (SRR) or Cross-Region Replication (CRR)—add a replication configuration to your source bucket. This configuration tells Amazon S3 to replicate objects as specified. In the replication configuration, you must provide the following:

- **The destination buckets** – The bucket or buckets where you want Amazon S3 to replicate the objects.
- **The objects that you want to replicate** – You can replicate all objects in the source bucket or a subset of objects. You identify a subset by providing a key name prefix, one or more object tags, or both in the configuration.

  For example, if you configure a replication rule to replicate only objects with the key name prefix `Tax/`, Amazon S3 replicates objects with keys such as `Tax/doc1` or `Tax/doc2`. But it doesn't

replicate objects with the key `Legal/doc3`. If you specify both a prefix and one or more tags, Amazon S3 replicates only objects that have the specific key prefix and tags.

- **An AWS Identity and Access Management (IAM) role** – Amazon S3 assumes this IAM role to replicate objects on your behalf. For more information about creating this IAM role and managing permissions, see Setting up permissions for live replication.

In addition to these minimum requirements, you can choose the following options:

- **Replica storage class** – By default, Amazon S3 stores object replicas using the same storage class as the source object. You can specify a different storage class for the replicas.
- **Replica ownership** – Amazon S3 assumes that an object replica continues to be owned by the owner of the source object. So when it replicates objects, it also replicates the corresponding object access control list (ACL) or S3 Object Ownership setting. If the source and destination buckets are owned by different AWS accounts, you can configure replication to change the owner of a replica to the AWS account that owns the destination bucket. For more information, see the section called “Changing the replica owner”.

You can configure replication by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or the Amazon S3 REST API. For detailed walkthroughs of how to set up replication, see the section called “Replication walkthroughs”.

Amazon S3 provides REST API operations to support setting up replication rules. For more information, see the following topics in the *Amazon Simple Storage Service API Reference*:

- PutBucketReplication
- GetBucketReplication
- DeleteBucketReplication

Topics

- Replication configuration file elements
- Setting up permissions for live replication
- Examples for configuring live replication

Replication configuration file elements

Amazon S3 stores a replication configuration as XML. If you're configuring replication programmatically through the Amazon S3 REST API, you specify the various elements of your replication configuration in this XML file. If you're configuring replication through the AWS Command Line Interface (AWS CLI), you specify your replication configuration using JSON format. For JSON examples, see the walkthroughs in the section called “Replication walkthroughs”.

> **Note**
>
> The latest version of the replication configuration XML format is V2. XML V2 replication configurations are those that contain the `<Filter>` element for rules, and rules that specify S3 Replication Time Control (S3 RTC).
>
> To see your replication configuration version, you can use the `GetBucketReplication` API operation. For more information, see GetBucketReplication in the *Amazon Simple Storage Service API Reference*.
>
> For backward compatibility, Amazon S3 continues to support the XML V1 replication configuration format. If you've used the XML V1 replication configuration format, see Backward compatibility considerations for backward compatibility considerations.

In the replication configuration XML file, you must specify an AWS Identity and Access Management (IAM) role and one or more rules, as shown in the following example:

```
<ReplicationConfiguration>
    <Role>IAM-role-ARN</Role>
    <Rule>
        ...
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
</ReplicationConfiguration>
```

Amazon S3 can't replicate objects without your permission. You grant permissions to Amazon S3 with the IAM role that you specify in the replication configuration. Amazon S3 assumes this IAM role to replicate objects on your behalf. You must grant the required permissions to the IAM

role first. For more information about managing permissions, see Setting up permissions for live replication.

An Amazon S3 replication configuration can have up to 1,000 rules per source bucket. This limit is not adjustable.

You add only one rule in a replication configuration in the following scenarios:

- You want to replicate all objects.
- You want to replicate only one subset of objects. You identify the object subset by adding a filter in the rule. In the filter, you specify an object key prefix, tags, or a combination of both to identify the subset of objects that the rule applies to. The filters target objects that match the exact values that you specify.

If you want to replicate different subsets of objects, you add multiple rules in a replication configuration. In each rule, you specify a filter that selects a different subset. For example, you might choose to replicate objects that have either `tax/` or `document/` key prefixes. To do this, you add 2 rules, one that specifies the `tax/` key prefix filter and another that specifies the `document/` key prefix. For more information about object key prefixes, see Organizing objects using prefixes.

The following sections provide additional information.

Topics

- Basic rule configuration
- Optional: Specifying a filter
- Additional destination configurations
- Example replication configurations
- Backward compatibility considerations

Basic rule configuration

Each rule must include the rule's status and priority. The rule must also indicate whether to replicate delete markers.

- The `<Status>` element indicates whether the rule is enabled or disabled by using the values `Enabled` or `Disabled`. If a rule is disabled, Amazon S3 doesn't perform the actions specified in the rule.

- The <Priority> element indicates which rule has precedence whenever 2 or more replication rules conflict. Amazon S3 attempts to replicate objects according to all replication rules. However, if there are 2 or more rules with the same destination bucket, then objects are replicated according to the rule with the highest priority. The higher the number, the higher the priority.
- The <DeleteMarkerReplication> element indicates whether to replicate delete markers by using the values Enabled or Disabled.

In the <Destination> element configuration, you must provide the name of the destination bucket or buckets where you want Amazon S3 to replicate objects.

The following example shows the minimum requirements for a V2 rule. For backward compatibility, Amazon S3 continues to support the XML V1 format. For more information, see Backward compatibility considerations.

```
...
    <Rule>
        <ID>Rule-1</ID>
        <Status>Enabled-or-Disabled</Status>
        <Filter>
            <Prefix></Prefix>
        </Filter>
        <Priority>integer</Priority>
        <DeleteMarkerReplication>
           <Status>Enabled-or-Disabled</Status>
        </DeleteMarkerReplication>
        <Destination>
           <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
        </Destination>
    </Rule>
    <Rule>
         ...
    </Rule>
     ...
...
```

You can also specify other configuration options. For example, you might choose to use a storage class for object replicas that differs from the class for the source object.

Optional: Specifying a filter

To choose a subset of objects that the rule applies to, add an optional filter. You can filter by object key prefix, object tags, or a combination of both. If you filter on both a key prefix and object tags, Amazon S3 combines the filters by using a logical AND operator. In other words, the rule applies to a subset of objects with both a specific key prefix and specific tags.

Filter based on object key prefix

To specify a rule with a filter based on an object key prefix, use the following XML. You can specify only one prefix per rule.

```
<Rule>
    ...
    <Filter>
        <Prefix>key-prefix</Prefix>
    </Filter>
    ...
</Rule>
...
```

Filter based on object tags

To specify a rule with a filter based on object tags, use the following XML. You can specify one or more object tags.

```
<Rule>
    ...
    <Filter>
        <And>
            <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
        </And>
    </Filter>
    ...
```

```
</Rule>
...
```

Filter with a key prefix and object tags

To specify a rule filter with a combination of a key prefix and object tags, use the following XML. You wrap these filters in an <And> parent element. Amazon S3 performs a logical AND operation to combine these filters. In other words, the rule applies to a subset of objects with both a specific key prefix and specific tags.

```
<Rule>
    ...
    <Filter>
        <And>
            <Prefix>key-prefix</Prefix>
            <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
            </Tag>
            <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
            </Tag>
             ...
    </Filter>
    ...
</Rule>
...
```

Note

- If you specify a rule with an empty <Filter> element, your rule applies to all objects in your bucket.
- When you're using tag-based replication rules with live replication, new objects must be tagged with the matching replication rule tag in the PutObject operation. Otherwise, the objects won't be replicated. If objects are tagged after the PutObject operation, those objects also won't be replicated.

To replicate objects that have been tagged after the `PutObject` operation, you must use S3 Batch Replication. For more information about Batch Replication, see Replicating existing objects.

Additional destination configurations

In the destination configuration, you specify the bucket or buckets where you want Amazon S3 to replicate objects. You can set configurations to replicate objects from one source bucket to one or more destination buckets.

```
...
<Destination>
    <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
</Destination>
...
```

You can add the following options in the `<Destination>` element.

Topics

- Specify storage class
- Add multiple destination buckets
- Specify different parameters for each replication rule with multiple destination buckets
- Change replica ownership
- Enable S3 Replication Time Control
- Replicate objects created with server-side encryption by using AWS KMS

Specify storage class

You can specify the storage class for the object replicas. By default, Amazon S3 uses the storage class of the source object to create object replicas, as in the following example.

```
...
<Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
       <StorageClass>storage-class</StorageClass>
</Destination>
```

```
...
```

Add multiple destination buckets

You can add multiple destination buckets in a single replication configuration, as follows.

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket1</Bucket>
    </Destination>
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled-or-Disabled</Status>
    </DeleteMarkerReplication>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket2</Bucket>
    </Destination>
</Rule>
...
```

Specify different parameters for each replication rule with multiple destination buckets

When adding multiple destination buckets in a single replication configuration, you can specify different parameters for each replication rule, as follows.

```
...
<Rule>
    <ID>Rule-1</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Disabled</Status>
```

```
    </DeleteMarkerReplication>
      <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket1</Bucket>
    </Destination>
</Rule>
<Rule>
    <ID>Rule-2</ID>
    <Status>Enabled-or-Disabled</Status>
    <Priority>integer</Priority>
    <DeleteMarkerReplication>
       <Status>Enabled</Status>
    </DeleteMarkerReplication>
      <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
  <ReplicationTime>
    <Status>Enabled</Status>
    <Time>
      <Minutes>15</Minutes>
    </Time>
  </ReplicationTime>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket2</Bucket>
    </Destination>
</Rule>
...
```

Change replica ownership

When the source and destination buckets aren't owned by the same accounts, you can change the ownership of the replica to the AWS account that owns the destination bucket. To do so, add the `<AccessControlTranslation>` element. This element takes the value `Destination`.

```
...
```

```
<Destination>
   <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
   <Account>destination-bucket-owner-account-id</Account>
   <AccessControlTranslation>
       <Owner>Destination</Owner>
   </AccessControlTranslation>
</Destination>
...
```

If you don't add the <AccessControlTranslation> element to the replication configuration, the replicas are owned by the same AWS account that owns the source object. For more information, see Changing the replica owner.

Enable S3 Replication Time Control

You can enable S3 Replication Time Control (S3 RTC) in your replication configuration. S3 RTC replicates most objects in seconds and 99.99 percent of objects within 15 minutes (backed by a service-level agreement).

> **Note**
>
> Only a value of <Minutes>15</Minutes> is accepted for the <EventThreshold> and <Time> elements.

```
...
<Destination>
  <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
  <Metrics>
    <Status>Enabled</Status>
    <EventThreshold>
      <Minutes>15</Minutes>
    </EventThreshold>
  </Metrics>
  <ReplicationTime>
    <Status>Enabled</Status>
    <Time>
      <Minutes>15</Minutes>
    </Time>
  </ReplicationTime>
</Destination>
```

```
...
```

For more information, see Meeting compliance requirements with S3 Replication Time Control. For API examples, see PutBucketReplication in the *Amazon Simple Storage Service API Reference*.

Replicate objects created with server-side encryption by using AWS KMS

Your source bucket might contain objects that were created with server-side encryption by using AWS Key Management Service (AWS KMS) keys (SSE-KMS). By default, Amazon S3 doesn't replicate these objects. You can optionally direct Amazon S3 to replicate these objects. To do so, first explicitly opt into this feature by adding the `<SourceSelectionCriteria>` element. Then provide the AWS KMS key (for the AWS Region of the destination bucket) to use for encrypting object replicas. The following example shows how to specify these elements.

```
...
<SourceSelectionCriteria>
  <SseKmsEncryptedObjects>
    <Status>Enabled</Status>
  </SseKmsEncryptedObjects>
</SourceSelectionCriteria>
<Destination>
  <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
  <EncryptionConfiguration>
    <ReplicaKmsKeyID>AWS KMS key ID to use for encrypting object replicas</
ReplicaKmsKeyID>
  </EncryptionConfiguration>
</Destination>
...
```

For more information, see Replicating encrypted objects (SSE-S3, SSE-KMS, DSSE-KMS, SSE-C).

Example replication configurations

To get started, you can add the following example replication configurations to your bucket, as appropriate.

> **Important**
>
> To add a replication configuration to a bucket, you must have the `iam:PassRole` permission. This permission allows you to pass the IAM role that grants Amazon S3 replication permissions. You specify the IAM role by providing the Amazon Resource Name

(ARN) that is used in the <Role> element in the replication configuration XML. For more information, see Granting a User Permissions to Pass a Role to an AWS service in the *IAM User Guide*.

Example 1: Replication configuration with one rule

The following basic replication configuration specifies one rule. The rule specifies an IAM role that Amazon S3 can assume and a single destination bucket for object replicas. The <Status> element value of Enabled indicates that the rule is in effect.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>

    <Destination>
      <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

To choose a subset of objects to replicate, you can add a filter. In the following configuration, the filter specifies an object key prefix. This rule applies to objects that have the prefix *Tax/* in their key names.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
       <Prefix>Tax/</Prefix>
    </Filter>
```

```
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
    </Destination>

  </Rule>
</ReplicationConfiguration>
```

If you specify the <Filter> element, you must also include the <Priority> and <DeleteMarkerReplication> elements. In this example, the value that you set for the <Priority> element is irrelevant because there is only one rule.

In the following configuration, the filter specifies 1 prefix and 2 tags. The rule applies to the subset of objects that have the specified key prefix and tags. Specifically, it applies to objects that have the *Tax/* prefix in their key names and the 2 specified object tags. In this example, the value that you set for the <Priority> element is irrelevant because there is only one rule.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>

    <Filter>
        <And>
          <Prefix>Tax/</Prefix>
          <Tag>
             <Tag>
                <Key>tagA</Key>
                <Value>valueA</Value>
             </Tag>
          </Tag>
          <Tag>
             <Tag>
                <Key>tagB</Key>
                <Value>valueB</Value>
             </Tag>
          </Tag>
       </And>
```

```
    </Filter>

    <Destination>
        <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
    </Destination>

  </Rule>
</ReplicationConfiguration>
```

You can specify a storage class for the object replicas as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
       <StorageClass>storage-class</StorageClass>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

You can specify any storage class that Amazon S3 supports.

Example 2: Replication configuration with 2 rules

Example

In the following replication configuration, the rules specify the following:

- Each rule filters on a different key prefix so that each rule applies to a distinct subset of objects. In this example, Amazon S3 replicates objects with the key names *Tax/doc1.pdf* and *Project/project1.txt*, but it doesn't replicate objects with the key name *PersonalDoc/documentA*.
- Although both rules specify a value for the `<Priority>` element, the rule priority is irrelevant because the rules apply to 2 distinct sets of objects. The next example shows what happens when rule priority is applied.

- The second rule specifies the S3 Standard-IA storage class for object replicas. Amazon S3 uses the specified storage class for those object replicas.

```
<?xml version="1.0" encoding="UTF-8"?>

<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Tax</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
    </Destination>
     ...
  </Rule>
 <Rule>
    <Status>Enabled</Status>
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>Project</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Destination>
      <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
     <StorageClass>STANDARD_IA</StorageClass>
    </Destination>
     ...
  </Rule>


</ReplicationConfiguration>
```

Example 3: Replication configuration with 2 rules with overlapping prefixes

In this configuration, the 2 rules specify filters with overlapping key prefixes, *star* and *starship*. Both rules apply to objects with the key name *starship-x*. In this case, Amazon S3 uses the rule priority to determine which rule to apply. The higher the number, the higher the priority.

```
<ReplicationConfiguration>

  <Role>arn:aws:iam::account-id:role/role-name</Role>

  <Rule>
    <Status>Enabled</Status>
    <Priority>1</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>star</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
    </Destination>
  </Rule>
  <Rule>
    <Status>Enabled</Status>
    <Priority>2</Priority>
    <DeleteMarkerReplication>
       <Status>string</Status>
    </DeleteMarkerReplication>
    <Filter>
        <Prefix>starship</Prefix>
    </Filter>
    <Destination>
      <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

Example 4: Example walkthroughs

For example walkthroughs, see Examples for configuring live replication.

For more information about the XML structure of replication configuration, see PutBucketReplication in the *Amazon Simple Storage Service API Reference*.

Backward compatibility considerations

The latest version of the replication configuration XML format is V2. XML V2 replication configurations are those that contain the `<Filter>` element for rules, and rules that specify S3 Replication Time Control (S3 RTC).

To see your replication configuration version, you can use the `GetBucketReplication` API operation. For more information, see GetBucketReplication in the *Amazon Simple Storage Service API Reference*.

For backward compatibility, Amazon S3 continues to support the XML V1 replication configuration format. If you've used the XML V1 replication configuration format, consider the following issues that affect backward compatibility:

- The replication configuration XML V2 format includes the `<Filter>` element for rules. With the `<Filter>` element, you can specify object filters based on the object key prefix, tags, or both to scope the objects that the rule applies to. The replication configuration XML V1 format supports filtering based only on the key prefix. In that case, you add the `<Prefix>` element directly as a child element of the `<Rule>` element, as in the following example:

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Role>arn:aws:iam::account-id:role/role-name</Role>
  <Rule>
    <Status>Enabled</Status>
    <Prefix>key-prefix</Prefix>
    <Destination>
       <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
    </Destination>

  </Rule>
</ReplicationConfiguration>
```

- When you delete an object from your source bucket without specifying an object version ID, Amazon S3 adds a delete marker. If you use the replication configuration XML V1 format, Amazon S3 replicates only delete markers that result from user actions. In other words, Amazon S3 replicates the delete marker only if a user deletes an object. If an expired object is removed by Amazon S3 (as part of a lifecycle action), Amazon S3 doesn't replicate the delete marker.

In the replication configuration XML V2 format, you can enable delete marker replication for non-tag-based rules. For more information, see Replicating delete markers between buckets.

Setting up permissions for live replication

When setting up live replication in Amazon S3, you must acquire the necessary permissions as follows:

- You must grant the AWS Identity and Access Management (IAM) principal (user or role) who will be creating replication rules a certain set of permissions.
- Amazon S3 needs permissions to replicate objects on your behalf. You grant these permissions by creating an IAM role and then specifying that role in your replication configuration.
- When the source and destination buckets aren't owned by the same accounts, the owner of the destination bucket must also grant the source bucket owner permissions to store the replicas.

> **Note**
>
> If you're using S3 Batch Operations to replicate objects on demand instead of setting up live replication, a different IAM role and policies are required for S3 Batch Replication. For a Batch Replication IAM role and policy examples, see Configuring an IAM role for S3 Batch Replication.

Topics

- Step 1: Granting permissions to the IAM principal who's creating replication rules
- Step 2: Creating an IAM role for Amazon S3 to assume
- (Optional) Step 3: Granting permissions when the source and destination buckets are owned by different AWS accounts
- (Optional) Step 4: Granting permissions to change replica ownership

Step 1: Granting permissions to the IAM principal who's creating replication rules

The IAM user or role that you will use to create replication rules needs permissions to create replication rules for one- or two-way replications. If the user or role doesn't have these

permissions, you won't be able to create replication rules. For more information, see IAM Identities in the *IAM User Guide*.

The user or role needs the following actions:

- `iam:AttachRolePolicy`
- `iam:CreatePolicy`
- `iam:CreateServiceLinkedRole`
- `iam:PassRole`
- `iam:PutRolePolicy`
- `s3:GetBucketVersioning`
- `s3:GetObjectVersionAcl`
- `s3:GetObjectVersionForReplication`
- `s3:GetReplicationConfiguration`
- `s3:PutReplicationConfiguration`

Following is a sample IAM policy that includes these actions.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetAccessPoint",
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets",
                "s3:PutReplicationConfiguration",
                "s3:GetReplicationConfiguration",
                "s3:GetBucketVersioning",
```

```
                "s3:GetObjectVersionForReplication",
                "s3:GetObjectVersionAcl",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetObjectVersion",
                "s3:GetBucketOwnershipControls",
                "s3:PutBucketOwnershipControls",
                "s3:GetObjectLegalHold",
                "s3:GetObjectRetention",
                "s3:GetBucketObjectLockConfiguration"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1-*",
                "arn:aws:s3:::amzn-s3-demo-bucket2-*/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:List*AccessPoint*",
                "s3:GetMultiRegion*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:Get*",
                "iam:CreateServiceLinkedRole",
                "iam:CreateRole",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::*:role/service-role/s3*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:List*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
```

```
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy",
                "iam:CreatePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:policy/service-role/s3*",
                "arn:aws:iam::*:role/service-role/s3*"
            ]
        }
    ]
}
```

Step 2: Creating an IAM role for Amazon S3 to assume

By default, all Amazon S3 resources—buckets, objects, and related subresources—are private, and only the resource owner can access the resource. Amazon S3 needs permissions to read and replicate objects from the source bucket. You grant these permissions by creating an IAM role and specifying that role in your replication configuration.

This section explains the trust policy and the minimum required permissions policy that are attached to this IAM role. The example walkthroughs provide step-by-step instructions to create an IAM role. For more information, see Examples for configuring live replication.

> **Note**
>
> If you're using the console to create your replication configuration, we recommend that you skip this section and instead have the console create this IAM role and the necessary trust and permission policies for you.

The *trust policy* identifies which principal identities can assume the IAM role. The *permissions policy* specifies which actions the IAM role can perform, on which resources, and under what conditions.

- The following example shows a *trust policy* where you identify Amazon S3 as the AWS service principal that can assume the role:

  JSON

```
{
```

```
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

- The following example shows a *trust policy* where you identify Amazon S3 and S3 Batch Operations as service principals that can assume the role. Use this approach if you're creating a Batch Replication job. For more information, see Create a Batch Replication job for new replication rules or destinations.

  JSON

  ```
  {
     "Version":"2012-10-17",
     "Statement":[
        {
           "Effect":"Allow",
           "Principal":{
              "Service": [
                "s3.amazonaws.com",
                "batchoperations.s3.amazonaws.com"
              ]
           },
           "Action":"sts:AssumeRole"
        }
     ]
  }
  ```

  For more information about IAM roles, see IAM roles in the *IAM User Guide*.

- The following example shows the *permissions policy*, where you grant the IAM role permissions to perform replication tasks on your behalf. When Amazon S3 assumes the role, it has the permissions that you specify in this policy. In this policy, *`amzn-s3-demo-source-bucket`* is the source bucket, and *`amzn-s3-demo-destination-bucket`* is the destination bucket.

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      }
   ]
}
```

The permissions policy grants permissions for the following actions:

- `s3:GetReplicationConfiguration` and `s3:ListBucket` – Permissions for these actions on the *amzn-s3-demo-source-bucket* bucket allow Amazon S3 to retrieve the replication

configuration and list the bucket content. (The current permissions model requires the `s3:ListBucket` permission for accessing delete markers.)

- `s3:GetObjectVersionForReplication` and `s3:GetObjectVersionAcl` – Permissions for these actions are granted on all objects to allow Amazon S3 to get a specific object version and access control list (ACL) associated with the objects.

- `s3:ReplicateObject` and `s3:ReplicateDelete` – Permissions for these actions on all objects in the *amzn-s3-demo-destination-bucket* bucket allow Amazon S3 to replicate objects or delete markers to the destination bucket. For information about delete markers, see How delete operations affect replication.

  > **Note**
  >
  > Permissions for the `s3:ReplicateObject` action on the *amzn-s3-demo-destination-bucket* bucket also allow replication of metadata such as object tags and ACLs. Therefore, you don't need to explicitly grant permission for the `s3:ReplicateTags` action.

- `s3:GetObjectVersionTagging` – Permissions for this action on objects in the *amzn-s3-demo-source-bucket* bucket allow Amazon S3 to read object tags for replication. For more information about object tags, see Tagging your objects. If Amazon S3 doesn't have the `s3:GetObjectVersionTagging` permission, it replicates the objects, but not the object tags.
- `s3:GetObjectVersionAnnotationForReplication` – Permissions for this action on objects in the source bucket allow Amazon S3 to read annotations for replication. If Amazon S3 doesn't have this permission, annotation replication fails. For more information about annotations, see Annotating your objects.

For a list of Amazon S3 actions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

> **Important**
>
> The AWS account that owns the IAM role must have permissions for the actions that it grants to the IAM role.

For example, suppose that the source bucket contains objects owned by another AWS account. The owner of the objects must explicitly grant the AWS account that owns the IAM role the required permissions through the objects' access control lists (ACLs). Otherwise, Amazon S3 can't access the objects, and replication of the objects fails. For information about ACL permissions, see Access control list (ACL) overview.
The permissions described here are related to the minimum replication configuration. If you choose to add optional replication configurations, you must grant additional permissions to Amazon S3:

- To replicate encrypted objects, you also need to grant the necessary AWS Key Management Service (AWS KMS) key permissions. For more information, see the section called "Replicating encrypted objects".
- To use Object Lock with replication, you must grant 2 additional permissions on the source S3 bucket in the AWS Identity and Access Management (IAM) role that you use to set up replication. The 2 additional permissions are `s3:GetObjectRetention` and `s3:GetObjectLegalHold`. If the role has an `s3:Get*` permission statement, that statement satisfies the requirement. For more information, see the section called "Using Object Lock with S3 Replication".

(Optional) Step 3: Granting permissions when the source and destination buckets are owned by different AWS accounts

When the source and destination buckets aren't owned by the same accounts, the owner of the destination bucket must also add a bucket policy to grant the owner of the source bucket permissions to perform replication actions, as shown in the following example. In this example policy, *amzn-s3-demo-destination-bucket* is the destination bucket.

You can also use the Amazon S3 console to automatically generate this bucket policy for you. For more information, see Enable receiving replicated objects from a source bucket.

Note

The ARN format of the role might appear different. If the role was created by using the console, the ARN format is `arn:aws:iam::`*account-ID*`:role/service-role/`*role-name*. If the role was created by using the AWS CLI, the ARN format is `arn:aws:iam::`*account-ID*`:role/`*role-name*. For more information, see IAM roles in the *IAM User Guide*.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "PolicyForDestinationBucket",
    "Statement": [
        {
            "Sid": "Permissions on objects",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/service-role/source-
account-IAM-role"
            },
            "Action": [
                "s3:ReplicateDelete",
                "s3:ReplicateObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
        },
        {
            "Sid": "Permissions on bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/service-role/source-
account-IAM-role"
            },
            "Action": [
                "s3:List*",
                "s3:GetBucketVersioning",
                "s3:PutBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket"
        }
    ]
}
```

For an example, see Configuring replication for buckets in different accounts.

If objects in the source bucket are tagged, note the following:

- If the source bucket owner grants Amazon S3 permission for the `s3:GetObjectVersionTagging` and `s3:ReplicateTags` actions to replicate object tags (through the IAM role), Amazon S3 replicates the tags along with the objects. For information about the IAM role, see Step 2: Creating an IAM role for Amazon S3 to assume.
- If the owner of the destination bucket doesn't want to replicate the tags, they can add the following statement to the destination bucket policy to explicitly deny permission for the `s3:ReplicateTags` action. In this policy, *amzn-s3-demo-destination-bucket* is the destination bucket.

```
...
   "Statement":[
      {
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-account-id:role/service-role/source-account-IAM-role"
         },
         "Action":"s3:ReplicateTags",
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      }
   ]
...
```

- Similarly, if the source bucket owner grants Amazon S3 permission for the `s3:GetObjectVersionAnnotationForReplication` action to replicate annotations (through the IAM role), Amazon S3 replicates the annotations along with the objects. For information about the IAM role, see Step 2: Creating an IAM role for Amazon S3 to assume.

  If the owner of the destination bucket doesn't want to replicate the annotations, they can add the following statement to the destination bucket policy to explicitly deny permission for the `s3:ReplicateObjectAnnotation` action. In this policy, *amzn-s3-demo-destination-bucket* is the destination bucket.

```
...
   "Statement":[
      {
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-account-id:role/service-role/source-account-IAM-role"
```

```
            },
            "Action":"s3:ReplicateObjectAnnotation",
            "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
        }
    ]
...
```

Note

- If you want to replicate encrypted objects, you also must grant the necessary AWS Key Management Service (AWS KMS) key permissions. For more information, see the section called "Replicating encrypted objects".
- To use Object Lock with replication, you must grant 2 additional permissions on the source S3 bucket in the AWS Identity and Access Management (IAM) role that you use to set up replication. The 2 additional permissions are `s3:GetObjectRetention` and `s3:GetObjectLegalHold`. If the role has an `s3:Get*` permission statement, that statement satisfies the requirement. For more information, see the section called "Using Object Lock with S3 Replication".

Enable receiving replicated objects from a source bucket

Instead of manually adding the preceding policy to your destination bucket, you can quickly generate the policies needed to enable receiving replicated objects from a source bucket through the Amazon S3 console.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the bucket that you want to use as a destination bucket.
4. Choose the **Management** tab, and scroll down to **Replication rules**.
5. For **Actions**, choose **Receive replicated objects**.

   Follow the prompts and enter the AWS account ID of the source bucket account, and then choose **Generate policies**. The console generates an Amazon S3 bucket policy and a KMS key policy.

6. To add this policy to your existing bucket policy, either choose **Apply settings** or choose **Copy** to manually copy the changes.
7. (Optional) Copy the AWS KMS policy to your desired KMS key policy in the AWS Key Management Service console.

(Optional) Step 4: Granting permissions to change replica ownership

When different AWS accounts own the source and destination buckets, you can tell Amazon S3 to change the ownership of the replica to the AWS account that owns the destination bucket. To override the ownership of replicas, you must either grant some additional permissions or adjust the S3 Object Ownership settings for the destination bucket. For more information about owner override, see Changing the replica owner.

Examples for configuring live replication

The following examples provide step-by-step walkthroughs that show how to configure live replication for common use cases.

> **Note**
>
> Live replication refers to Same-Region Replication (SRR) and Cross-Region Replication (CRR). Live replication doesn't replicate any objects that existed in the bucket before you set up replication. To replicate objects that existed before you set up replication, use on-demand replication. To sync buckets and replicate existing objects on demand, see Replicating existing objects.

These examples demonstrate how to create a replication configuration by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDKs (AWS SDK for Java and AWS SDK for .NET examples are shown).

For information about installing and configuring the AWS CLI, see the following topics in the *AWS Command Line Interface User Guide*:

- Get started with the AWS CLI
- Configure the AWS CLI – You must set up at least one profile. If you are exploring cross-account scenarios, set up two profiles.

For information about the AWS SDKs, see AWS SDK for Java and AWS SDK for .NET.

> **Tip**
>
> For a step-by-step tutorial that demonstrates how to use live replication to replicate data, see Tutorial: Replicating data within and between AWS Regions using S3 Replication.

Topics

- Configuring replication for buckets in the same account
- Configuring replication for buckets in different accounts
- Meeting compliance requirements with S3 Replication Time Control
- Replicating encrypted objects (SSE-S3, SSE-KMS, DSSE-KMS, SSE-C)
- Replicating metadata changes with replica modification sync
- Replicating delete markers between buckets
- Replicating annotations

Configuring replication for buckets in the same account

Live replication is the automatic, asynchronous copying of objects across general purpose buckets in the same or different AWS Regions. Live replication copies newly created objects and object updates from a source bucket to a destination bucket or buckets. For more information, see Replicating objects within and across Regions.

When you configure replication, you add replication rules to the source bucket. Replication rules define which source bucket objects to replicate and the destination bucket or buckets where the replicated objects are stored. You can create a rule to replicate all the objects in a bucket or a subset of objects with a specific key name prefix, one or more object tags, or both. A destination bucket can be in the same AWS account as the source bucket, or it can be in a different account.

If you specify an object version ID to delete, Amazon S3 deletes that object version in the source bucket. But it doesn't replicate the deletion in the destination bucket. In other words, it doesn't delete the same object version from the destination bucket. This protects data from malicious deletions.

When you add a replication rule to a bucket, the rule is enabled by default, so it starts working as soon as you save it.

In this example, you set up live replication for source and destination buckets that are owned by the same AWS account. Examples are provided for using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), and the AWS SDK for Java and AWS SDK for .NET.

Prerequisites

Before you use the following procedures, make sure that you've set up the necessary permissions for replication, depending on whether the source and destination buckets are owned by the same or different accounts. For more information, see the section called "Setting up permissions".

Note

- If you want to replicate encrypted objects, you also must grant the necessary AWS Key Management Service (AWS KMS) key permissions. For more information, see the section called "Replicating encrypted objects".
- To use Object Lock with replication, you must grant two additional permissions on the source S3 bucket in the AWS Identity and Access Management (IAM) role that you use to set up replication. The two additional permissions are `s3:GetObjectRetention` and `s3:GetObjectLegalHold`. If the role has an `s3:Get*` permission statement, that statement satisfies the requirement. For more information, see the section called "Using Object Lock with S3 Replication".

Using the S3 console

To configure a replication rule when the destination bucket is in the same AWS account as the source bucket, follow these steps.

If the destination bucket is in a different account from the source bucket, you must add a bucket policy to the destination bucket to grant the owner of the source bucket account permission to replicate objects in the destination bucket. For more information, see (Optional) Step 3: Granting permissions when the source and destination buckets are owned by different AWS accounts.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want.

4. Choose the **Management** tab, scroll down to **Replication rules**, and then choose **Create replication rule**.

5. In the **Replication rule configuration** section, under **Replication rule name**, enter a name for your rule to help identify the rule later. The name is required and must be unique within the bucket.

6. Under **Status**, **Enabled** is selected by default. An enabled rule starts to work as soon as you save it. If you want to enable the rule later, choose **Disabled**.

7. If the bucket has existing replication rules, you are instructed to set a priority for the rule. You must set a priority for the rule to avoid conflicts caused by objects that are included in the scope of more than one rule. In the case of overlapping rules, Amazon S3 uses the rule priority to determine which rule to apply. The higher the number, the higher the priority. For more information about rule priority, see Replication configuration file elements.

8. Under **Source bucket**, you have the following options for setting the replication source:

   - To replicate the whole bucket, choose **Apply to all objects in the bucket**.
   - To replicate all objects that have the same prefix, choose **Limit the scope of this rule using one or more filters**. This limits replication to all objects that have names that begin with the prefix that you specify (for example `pictures`). Enter a prefix in the **Prefix** box.

     > **Note**
     >
     > If you enter a prefix that is the name of a folder, you must use **/** (forward slash) as the last character (for example, `pictures/`).

   - To replicate all objects with one or more object tags, choose **Add tag** and enter the key-value pair in the boxes. Repeat the procedure to add another tag. You can combine a prefix and tags. For more information about object tags, see Tagging your objects.

   The new replication configuration XML schema supports prefix and tag filtering and the prioritization of rules. For more information about the new schema, see Backward compatibility considerations. For more information about the XML used with the Amazon S3 API that works behind the user interface, see Replication configuration file elements. The new schema is described as *replication configuration XML V2*.

9. Under **Destination**, choose the bucket where you want Amazon S3 to replicate objects.

> **Note**
>
> The number of destination buckets is limited to the number of AWS Regions in a given partition. A partition is a grouping of Regions. AWS currently has three partitions: `aws` (Standard Regions), `aws-cn` (China Regions), and `aws-us-gov` (AWS GovCloud (US) Regions). To request an increase in your destination bucket quota, you can use service quotas.

- To replicate to a bucket or buckets in your account, choose **Choose a bucket in this account**, and enter or browse for the destination bucket names.
- To replicate to a bucket or buckets in a different AWS account, choose **Specify a bucket in another account**, and enter the destination bucket account ID and bucket name.

  If the destination is in a different account from the source bucket, you must add a bucket policy to the destination buckets to grant the owner of the source bucket account permission to replicate objects. For more information, see (Optional) Step 3: Granting permissions when the source and destination buckets are owned by different AWS accounts.

  Optionally, if you want to help standardize ownership of new objects in the destination bucket, choose **Change object ownership to the destination bucket owner**. For more information about this option, see Controlling ownership of objects and disabling ACLs for your bucket.

> **Note**
>
> If versioning is not enabled on the destination bucket, you get a warning that contains an **Enable versioning** button. Choose this button to enable versioning on the bucket.

10. Set up an AWS Identity and Access Management (IAM) role that Amazon S3 can assume to replicate objects on your behalf.

    To set up an IAM role, in the **IAM role** section, select one of the following from the **IAM role** dropdown list:

- We highly recommend that you choose **Create new role** to have Amazon S3 create a new IAM role for you. When you save the rule, a new policy is generated for the IAM role that matches the source and destination buckets that you choose.
- You can choose to use an existing IAM role. If you do, you must choose a role that grants Amazon S3 the necessary permissions for replication. Replication fails if this role does not grant Amazon S3 sufficient permissions to follow your replication rule.

> **Important**
>
> When you add a replication rule to a bucket, you must have the `iam:PassRole` permission to be able to pass the IAM role that grants Amazon S3 replication permissions. For more information, see Granting a user permissions to pass a role to an AWS service in the *IAM User Guide*.

11. To replicate objects in the source bucket that are encrypted with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), under **Encryption**, select **Replicate objects encrypted with AWS KMS**. Under **AWS KMS keys for encrypting destination objects** are the source keys that you allow replication to use. All source KMS keys are included by default. To narrow the KMS key selection, you can choose an alias or key ID.

    Objects encrypted by AWS KMS keys that you do not select are not replicated. A KMS key or a group of KMS keys is chosen for you, but you can choose the KMS keys if you want. For information about using AWS KMS with replication, see Replicating encrypted objects (SSE-S3, SSE-KMS, DSSE-KMS, SSE-C).

    > **Important**
    >
    > When you replicate objects that are encrypted with AWS KMS, the AWS KMS request rate doubles in the source Region and increases in the destination Region by the same amount. These increased call rates to AWS KMS are due to the way that data is re-encrypted by using the KMS key that you define for the replication destination Region. AWS KMS has a request rate quota that is per calling account per Region. For information about the quota defaults, see AWS KMS Quotas - Requests per Second: Varies in the *AWS Key Management Service Developer Guide*.
    > If your current Amazon S3 PUT object request rate during replication is more than half the default AWS KMS rate limit for your account, we recommend that you request an

increase to your AWS KMS request rate quota. To request an increase, create a case in the Support Center at Contact Us. For example, suppose that your current PUT object request rate is 1,000 requests per second and you use AWS KMS to encrypt your objects. In this case, we recommend that you ask Support to increase your AWS KMS rate limit to 2,500 requests per second, in both your source and destination Regions (if different), to ensure that there is no throttling by AWS KMS.
To see your PUT object request rate in the source bucket, view `PutRequests` in the Amazon CloudWatch request metrics for Amazon S3. For information about viewing CloudWatch metrics, see Using the S3 console.

If you chose to replicate objects encrypted with AWS KMS, do the following:

- Under **AWS KMS key for encrypting destination objects** , specify your KMS key in one of the following ways:
  - To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose your **KMS key** from the list of available keys.

    Both the AWS managed key (`aws/s3`) and your customer managed keys appear in this list. For more information about customer managed keys, see Customer keys and AWS keys in the *AWS Key Management Service Developer Guide*.
  - To enter the KMS key Amazon Resource Name (ARN), choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears. This encrypts the replicas in the destination bucket. You can find the ARN for your KMS key in the IAM Console, under **Encryption keys**.
  - To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

    For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*.

    Important

    You can only use KMS keys that are enabled in the same AWS Region as the bucket. When you choose **Choose from your KMS keys**, the S3 console lists only 100 KMS keys per Region. If you have more than 100 KMS keys in the same

Region, you can see only the first 100 KMS keys in the S3 console. To use a KMS key that is not listed in the console, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN.
When you use an AWS KMS key for server-side encryption in Amazon S3, you must choose a symmetric encryption KMS key. Amazon S3 supports only symmetric encryption KMS keys and not asymmetric KMS keys. For more information, see Identifying symmetric and asymmetric KMS keys in the *AWS Key Management Service Developer Guide*.

For more information about creating an AWS KMS key, see Creating keys in the *AWS Key Management Service Developer Guide*. For more information about using AWS KMS with Amazon S3, see Using server-side encryption with AWS KMS keys (SSE-KMS).

12. Under **Destination storage class**, if you want to replicate your data into a specific storage class in the destination, choose **Change the storage class for the replicated objects**. Then choose the storage class that you want to use for the replicated objects in the destination. If you don't choose this option, the storage class for replicated objects is the same class as the original objects.
13. You have the following additional options while setting the **Additional replication options**:

    - If you want to enable S3 Replication Time Control (S3 RTC) in your replication configuration, select **Replication Time Control (RTC)**. For more information about this option, see Meeting compliance requirements with S3 Replication Time Control.
    - If you want to enable S3 Replication metrics in your replication configuration, select **Replication metrics and events**. For more information, see Monitoring replication with metrics, event notifications, and statuses.
    - If you want to enable delete marker replication in your replication configuration, select **Delete marker replication**. For more information, see Replicating delete markers between buckets.
    - If you want to enable Amazon S3 replica modification sync in your replication configuration, select **Replica modification sync**. For more information, see Replicating metadata changes with replica modification sync.

> **Note**
>
> When you use S3 RTC or S3 Replication metrics, additional fees apply.

14. To finish, choose **Save**.
15. After you save your rule, you can edit, enable, disable, or delete your rule by selecting your rule and choosing **Edit rule**.

Using the AWS CLI

To use the AWS CLI to set up replication when the source and destination buckets are owned by the same AWS account, you do the following:

- Create source and destination buckets.
- Enable versioning on the buckets.
- Create an AWS Identity and Access Management (IAM) role that gives Amazon S3 permission to replicate objects.
- Add the replication configuration to the source bucket.

To verify your setup, you test it.

To set up replication when the source and destination buckets are owned by the same AWS account

1. Set a credentials profile for the AWS CLI. This example uses the profile name `acctA`. For information about setting credential profiles and using named profiles, see Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

   > **Important**
   >
   > The profile that you use for this example must have the necessary permissions. For example, in the replication configuration, you specify the IAM role that Amazon S3 can assume. You can do this only if the profile that you use has the `iam:PassRole` permission. For more information, see Grant a user permissions to pass a role to an

AWS service in the *IAM User Guide*. If you use administrator credentials to create a named profile, you can perform all the tasks.

2. Create a source bucket and enable versioning on it by using the following AWS CLI commands. To use these commands, replace the *`user input placeholders`* with your own information.

   The following `create-bucket` command creates a source bucket named *`amzn-s3-demo-source-bucket`* in the US East (N. Virginia) (`us-east-1`) Region:

```
aws s3api create-bucket \
--bucket amzn-s3-demo-source-bucket \
--region us-east-1 \
--profile acctA
```

   The following `put-bucket-versioning` command enables S3 Versioning on the *`amzn-s3-demo-source-bucket`* bucket:

```
aws s3api put-bucket-versioning \
--bucket amzn-s3-demo-source-bucket \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. Create a destination bucket and enable versioning on it by using the following AWS CLI commands. To use these commands, replace the *`user input placeholders`* with your own information.

   Note

   To set up a replication configuration when both source and destination buckets are in the same AWS account, you use the same profile for the source and destination buckets. This example uses `acctA`.
   To test a replication configuration when the buckets are owned by different AWS accounts, specify different profiles for each account. For example, use an `acctB` profile for the destination bucket.

The following `create-bucket` command creates a destination bucket named *amzn-s3-demo-destination-bucket* in the US West (Oregon) (`us-west-2`) Region:

```
aws s3api create-bucket \
--bucket amzn-s3-demo-destination-bucket \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

The following `put-bucket-versioning` command enables S3 Versioning on the *amzn-s3-demo-destination-bucket* bucket:

```
aws s3api put-bucket-versioning \
--bucket amzn-s3-demo-destination-bucket \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. Create an IAM role. You specify this role in the replication configuration that you add to the source bucket later. Amazon S3 assumes this role to replicate objects on your behalf. You create an IAM role in two steps:

   - Create a role.
   - Attach a permissions policy to the role.

   a. Create the IAM role.

      i. Copy the following trust policy and save it to a file named `s3-role-trust-policy.json` in the current directory on your local computer. This policy grants the Amazon S3 service principal permissions to assume the role.

      JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
```

```
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

ii. Run the following command to create a role.

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
--profile acctA
```

b. Attach a permissions policy to the role.

i. Copy the following permissions policy and save it to a file named `s3-role-permissions-policy.json` in the current directory on your local computer. This policy grants permissions for various Amazon S3 bucket and object actions.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
```

```
            "Resource":[
               "arn:aws:s3:::amzn-s3-demo-source-bucket"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:ReplicateObject",
               "s3:ReplicateDelete",
               "s3:ReplicateTags"
            ],
            "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
         }
      ]
}
```

> **Note**
>
> - If you want to replicate encrypted objects, you also must grant the necessary AWS Key Management Service (AWS KMS) key permissions. For more information, see the section called "Replicating encrypted objects".
> - To use Object Lock with replication, you must grant two additional permissions on the source S3 bucket in the AWS Identity and Access Management (IAM) role that you use to set up replication. The two additional permissions are `s3:GetObjectRetention` and `s3:GetObjectLegalHold`. If the role has an `s3:Get*` permission statement, that statement satisfies the requirement. For more information, see the section called "Using Object Lock with S3 Replication".
> - To replicate object annotations, you must also grant `s3:GetObjectVersionAnnotationForReplication` on the source S3 bucket in the IAM role permissions policy. For more information, see the section called "Replicating metadata changes".

ii. Run the following command to create a policy and attach it to the role. Replace the *`user input placeholders`* with your own information.

```
$ aws iam put-role-policy \
--role-name replicationRole \
```

```
--policy-document file://s3-role-permissions-policy.json \
--policy-name replicationRolePolicy \
--profile acctA
```

5. Add a replication configuration to the source bucket.

   a. Although the Amazon S3 API requires that you specify the replication configuration as XML, the AWS CLI requires that you specify the replication configuration as JSON. Save the following JSON in a file called `replication.json` to the local directory on your computer.

```
{
  "Role": "IAM-role-ARN",
  "Rules": [
    {
      "Status": "Enabled",
      "Priority": 1,
      "DeleteMarkerReplication": { "Status": "Disabled" },
      "Filter" : { "Prefix": "Tax"},
      "Destination": {
        "Bucket": "arn:aws:s3:::amzn-s3-demo-destination-bucket"
      }
    }
  ]
}
```

   b. Update the JSON by replacing the values for the *amzn-s3-demo-destination-bucket* and *IAM-role-ARN* with your own information. Save the changes.

   c. Run the following `put-bucket-replication` command to add the replication configuration to your source bucket. Be sure to provide the source bucket name:

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket amzn-s3-demo-source-bucket \
--profile acctA
```

   To retrieve the replication configuration, use the `get-bucket-replication` command:

```
$ aws s3api get-bucket-replication \
--bucket amzn-s3-demo-source-bucket \
```

```
--profile acctA
```

6. Test the setup in the Amazon S3 console, by doing the following steps:

   a. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   b. In the left navigation pane, choose **Buckets**. In the **General purpose buckets** list, choose the source bucket.

   c. In the source bucket, create a folder named *Tax*.

   d. Add sample objects to the *Tax* folder in the source bucket.

   > **Note**
   >
   > The amount of time that it takes for Amazon S3 to replicate an object depends on the size of the object. For information about how to see the status of replication, see Getting replication status information.

   In the destination bucket, verify the following:

   - That Amazon S3 replicated the objects.
   - That the objects are replicas. On the **Properties** tab for your objects, scroll down to the **Object management overview** section. Under **Management configurations**, see the value under **Replication status**. Make sure that this value is set to `REPLICA`.
   - That the replicas are owned by the source bucket account. You can verify the object ownership on the **Permissions** tab for your objects.

     If the source and destination buckets are owned by different accounts, you can add an optional configuration to tell Amazon S3 to change the replica ownership to the destination account. For an example, see How to change the replica owner.

Using the AWS SDKs

Use the following code examples to add a replication configuration to a bucket with the AWS SDK for Java and AWS SDK for .NET, respectively.

Note

- If you want to replicate encrypted objects, you also must grant the necessary AWS Key Management Service (AWS KMS) key permissions. For more information, see the section called "Replicating encrypted objects".
- To use Object Lock with replication, you must grant two additional permissions on the source S3 bucket in the AWS Identity and Access Management (IAM) role that you use to set up replication. The two additional permissions are `s3:GetObjectRetention` and `s3:GetObjectLegalHold`. If the role has an `s3:Get*` permission statement, that statement satisfies the requirement. For more information, see the section called "Using Object Lock with S3 Replication".

Java

To add a replication configuration to a bucket and then retrieve and verify the configuration using the AWS SDK for Java, you can use the S3Client to manage replication settings programmatically.

For examples of how to configure replication with the AWS SDK for Java, see Set replication configuration on a bucket in the *Amazon S3 API Reference*.

C#

The following AWS SDK for .NET code example adds a replication configuration to a bucket and then retrieves it. To use this code, provide the names for your buckets and the Amazon Resource Name (ARN) for your IAM role. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CrossRegionReplicationTest
    {
```

```
        private const string sourceBucket = "*** source bucket ***";
        // Bucket ARN example - arn:aws:s3:::destinationbucket
        private const string destinationBucketArn = "*** destination bucket ARN
***";
        private const string roleArn = "*** IAM Role ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint sourceBucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;
        public static void Main()
        {
            s3Client = new AmazonS3Client(sourceBucketRegion);
            EnableReplicationAsync().Wait();
        }
        static async Task EnableReplicationAsync()
        {
            try
            {
                ReplicationConfiguration replConfig = new ReplicationConfiguration
                {
                    Role = roleArn,
                    Rules =
                        {
                            new ReplicationRule
                            {
                                Prefix = "Tax",
                                Status = ReplicationRuleStatus.Enabled,
                                Destination = new ReplicationDestination
                                {
                                    BucketArn = destinationBucketArn
                                }
                            }
                        }
                };

                PutBucketReplicationRequest putRequest = new
PutBucketReplicationRequest
                {
                    BucketName = sourceBucket,
                    Configuration = replConfig
                };

                PutBucketReplicationResponse putResponse = await
s3Client.PutBucketReplicationAsync(putRequest);
```

```
                // Verify configuration by retrieving it.
                await RetrieveReplicationConfigurationAsync(s3Client);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
        private static async Task RetrieveReplicationConfigurationAsync(IAmazonS3
 client)
        {
            // Retrieve the configuration.
            GetBucketReplicationRequest getRequest = new GetBucketReplicationRequest
            {
                BucketName = sourceBucket
            };
            GetBucketReplicationResponse getResponse = await
 client.GetBucketReplicationAsync(getRequest);
            // Print.
            Console.WriteLine("Printing replication configuration information...");
            Console.WriteLine("Role ARN: {0}", getResponse.Configuration.Role);
            foreach (var rule in getResponse.Configuration.Rules)
            {
                Console.WriteLine("ID: {0}", rule.Id);
                Console.WriteLine("Prefix: {0}", rule.Prefix);
                Console.WriteLine("Status: {0}", rule.Status);
            }
        }
    }
}
```

Configuring replication for buckets in different accounts

Live replication is the automatic, asynchronous copying of objects across buckets in the same or different AWS Regions. Live replication copies newly created objects and object updates from a

source bucket to a destination bucket or buckets. For more information, see Replicating objects within and across Regions.

When you configure replication, you add replication rules to the source bucket. Replication rules define which source bucket objects to replicate and the destination bucket or buckets where the replicated objects are stored. You can create a rule to replicate all the objects in a bucket or a subset of objects with a specific key name prefix, one or more object tags, or both. A destination bucket can be in the same AWS account as the source bucket, or it can be in a different account.

If you specify an object version ID to delete, Amazon S3 deletes that object version in the source bucket. But it doesn't replicate the deletion in the destination bucket. In other words, it doesn't delete the same object version from the destination bucket. This protects data from malicious deletions.

When you add a replication rule to a bucket, the rule is enabled by default, so it starts working as soon as you save it.

Setting up live replication when the source and destination buckets are owned by different AWS accounts is similar to setting up replication when both buckets are owned by the same account. However, there are several differences when you're configuring replication in a cross-account scenario:

- The destination bucket owner must grant the source bucket owner permission to replicate objects in the destination bucket policy.
- If you're replicating objects that are encrypted with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) in a cross-account scenario, the owner of the KMS key must grant the source bucket owner permission to use the KMS key. For more information, see Granting additional permissions for cross-account scenarios.
- By default, replicated objects are owned by the source bucket owner. In a cross-account scenario, you might want to configure replication to change the ownership of the replicated objects to the owner of the destination bucket. For more information, see Changing the replica owner.

To configure replication when the source and destination buckets are owned by different AWS accounts

1. In this example, you create source and destination buckets in two different AWS accounts. You must have two credential profiles set for the AWS CLI. This example uses `acctA` and `acctB` for those profile names. For information about setting credential profiles and using named

profiles, see Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

2. Follow the step-by-step instructions in Configuring replication for buckets in the same account with the following changes:

   - For all AWS CLI commands related to source bucket activities (such as creating the source bucket, enabling versioning, and creating the IAM role), use the `acctA` profile. Use the `acctB` profile to create the destination bucket.
   - Make sure that the permissions policy for the IAM role specifies the source and destination buckets that you created for this example.

3. In the console, add the following bucket policy on the destination bucket to allow the owner of the source bucket to replicate objects. For instructions, see Adding a bucket policy by using the Amazon S3 console. Be sure to edit the policy by providing the AWS account ID of the source bucket owner, the IAM role name, and the destination bucket name.

   Note

   To use the following example, replace the *`user input placeholders`* with your own information. Replace *`amzn-s3-demo-destination-bucket`* with your destination bucket name. Replace *`source-bucket-account-ID`*`:role/`*`source-account-IAM-role`* in the IAM Amazon Resource Name (ARN) with the IAM role that you're using for this replication configuration.
   The role path in the IAM ARN depends on how you created the role. If you created the role with the AWS CLI (as shown in Configuring replication for buckets in the same account), the path is `role/`*`role-name`*. If you created the role with the IAM console as a service role, the path includes `service-role/` (that is, `role/service-role/`*`role-name`*). For more information, see IAM ARNs in the *IAM User Guide*.

```
{
    "Version":"2012-10-17",
    "Id": "",
    "Statement": [
        {
            "Sid": "Set-permissions-for-objects",
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "arn:aws:iam::source-bucket-account-ID:role/source-account-
IAM-role"
            },
            "Action": [
                "s3:ReplicateObject",
                "s3:ReplicateDelete"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
        },
        {
            "Sid": "Set-permissions-on-bucket",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::source-bucket-account-ID:role/source-account-
IAM-role"
            },
            "Action": [
                "s3:GetBucketVersioning",
                "s3:PutBucketVersioning"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket"
        }
    ]
}
```

4. (Optional) If you're replicating objects that are encrypted with SSE-KMS, the owner of the KMS key must grant the source bucket owner permission to use the KMS key. For more information, see Granting additional permissions for cross-account scenarios.
5. (Optional) In replication, the owner of the source object owns the replica by default. When the source and destination buckets are owned by different AWS accounts, you can add optional configuration settings to change replica ownership to the AWS account that owns the destination buckets. This includes granting the `ObjectOwnerOverrideToBucketOwner` permission. For more information, see Changing the replica owner.

Changing the replica owner

In replication, the owner of the source object also owns the replica by default. However, when the source and destination buckets are owned by different AWS accounts, you might want to change the replica ownership. For example, you might want to change the ownership to restrict access to object replicas. In your replication configuration, you can add optional configuration settings to change replica ownership to the AWS account that owns the destination buckets.

To change the replica owner, you do the following:

- Add the *owner override* option to the replication configuration to tell Amazon S3 to change replica ownership.
- Grant Amazon S3 the `s3:ObjectOwnerOverrideToBucketOwner` permission to change replica ownership.
- Add the `s3:ObjectOwnerOverrideToBucketOwner` permission in the destination bucket policy to allow changing replica ownership. The `s3:ObjectOwnerOverrideToBucketOwner` permission allows the owner of the destination buckets to accept the ownership of object replicas.

For more information, see the section called "Considerations for the ownership override option" and Adding the owner override option to the replication configuration. For a working example with step-by-step instructions, see How to change the replica owner.

> **Important**
>
> Instead of using the owner override option, you can use the bucket owner enforced setting for Object Ownership. When you use replication and the source and destination buckets are owned by different AWS accounts, the bucket owner of the destination bucket can use the bucket owner enforced setting for Object Ownership to change replica ownership to the AWS account that owns the destination bucket. This setting disables object access control lists (ACLs).
>
> The bucket owner enforced setting mimics the existing owner override behavior without the need of the `s3:ObjectOwnerOverrideToBucketOwner` permission. All objects that are replicated to the destination bucket with the bucket owner enforced setting are owned by the destination bucket owner. For more information about Object Ownership, see Controlling ownership of objects and disabling ACLs for your bucket.

Considerations for the ownership override option

When you configure the ownership override option, the following considerations apply:

- By default, the owner of the source object also owns the replica. Amazon S3 replicates the object version and the ACL associated with it.

If you add the owner override option to your replication configuration, Amazon S3 replicates only the object version, not the ACL. In addition, Amazon S3 doesn't replicate subsequent changes to the source object ACL. Amazon S3 sets the ACL on the replica that grants full control to the destination bucket owner.

- When you update a replication configuration to enable or disable the owner override, the following behavior occurs:
  - If you add the owner override option to the replication configuration:

    When Amazon S3 replicates an object version, it discards the ACL that's associated with the source object. Instead, Amazon S3 sets the ACL on the replica, giving full control to the owner of the destination bucket. Amazon S3 doesn't replicate subsequent changes to the source object ACL. However, this ACL change doesn't apply to object versions that were replicated before you set the owner override option. ACL updates on source objects that were replicated before the owner override was set continue to be replicated (because the object and its replicas continue to have the same owner).
  - If you remove the owner override option from the replication configuration:

    Amazon S3 replicates new objects that appear in the source bucket and the associated ACLs to the destination buckets. For objects that were replicated before you removed the owner override, Amazon S3 doesn't replicate the ACLs because the object ownership change that Amazon S3 made remains in effect. That is, ACLs put on the object version that were replicated when the owner override was set continue to be not replicated.

Adding the owner override option to the replication configuration

> **Warning**
>
> Add the owner override option only when the source and destination buckets are owned by different AWS accounts. Amazon S3 doesn't check if the buckets are owned by the same or different accounts. If you add the owner override when both buckets are owned by same AWS account, Amazon S3 applies the owner override. This option grants full permissions to the owner of the destination bucket and doesn't replicate subsequent updates to the source objects' access control lists (ACLs). The replica owner can directly change the ACL associated with a replica with a `PutObjectAcl` request, but not through replication.

To specify the owner override option, add the following to each `Destination` element:

- The `AccessControlTranslation` element, which tells Amazon S3 to change replica ownership
- The `Account` element, which specifies the AWS account of the destination bucket owner

```
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    ...
    <Destination>
      ...
      <AccessControlTranslation>
           <Owner>Destination</Owner>
       </AccessControlTranslation>
      <Account>destination-bucket-owner-account-id</Account>
    </Destination>
  </Rule>
</ReplicationConfiguration>
```

The following example replication configuration tells Amazon S3 to replicate objects that have the *Tax* key prefix to the *amzn-s3-demo-destination-bucket* destination bucket and change ownership of the replicas. To use this example, replace the *user input placeholders* with your own information.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
         <Account>destination-bucket-owner-account-id</Account>
         <AccessControlTranslation>
            <Owner>Destination</Owner>
```

```
        </AccessControlTranslation>
      </Destination>
   </Rule>
</ReplicationConfiguration>
```

Granting Amazon S3 permission to change replica ownership

Grant Amazon S3 permissions to change replica ownership by adding permission for the `s3:ObjectOwnerOverrideToBucketOwner` action in the permissions policy that's associated with the AWS Identity and Access Management (IAM) role. This role is the IAM role that you specified in the replication configuration that allows Amazon S3 to assume and replicate objects on your behalf. To use the following example, replace *amzn-s3-demo-destination-bucket* with the name of the destination bucket.

```
...
{
    "Effect":"Allow",
         "Action":[
       "s3:ObjectOwnerOverrideToBucketOwner"
    ],
    "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
}
...
```

Adding permission in the destination bucket policy to allow changing replica ownership

The owner of the destination bucket must grant the owner of the source bucket permission to change replica ownership. The owner of the destination bucket grants the owner of the source bucket permission for the `s3:ObjectOwnerOverrideToBucketOwner` action. This permission allows the destination bucket owner to accept ownership of the object replicas. The following example bucket policy statement shows how to do this. To use this example, replace the *user input placeholders* with your own information.

```
...
{
    "Sid":"1",
    "Effect":"Allow",
    "Principal":{"AWS":"source-bucket-account-id"},
    "Action":["s3:ObjectOwnerOverrideToBucketOwner"],
    "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
```

```
}
...
```

How to change the replica owner

When the source and destination buckets in a replication configuration are owned by different AWS accounts, you can tell Amazon S3 to change replica ownership to the AWS account that owns the destination bucket. The following examples show how to use the Amazon S3 console, the AWS Command Line Interface (AWS CLI), and the AWS SDKs to change replica ownership.

Using the S3 console

For step-by-step instructions, see Configuring replication for buckets in the same account. This topic provides instructions for setting up a replication configuration when the source and destination buckets are owned by the same and different AWS accounts.

Using the AWS CLI

The following procedure shows how to change replica ownership by using the AWS CLI. In this procedure, you do the following:

- Create the source and destination buckets.
- Enable versioning on the buckets.
- Create an AWS Identity and Access Management (IAM) role that gives Amazon S3 permission to replicate objects.
- Add the replication configuration to the source bucket.
- In the replication configuration, you direct Amazon S3 to change the replica ownership.
- You test your replication configuration.

To change replica ownership when the source and destination buckets are owned by different AWS accounts (AWS CLI)

To use the example AWS CLI commands in this procedure, replace the *`user input placeholders`* with your own information.

1. In this example, you create the source and destination buckets in two different AWS accounts. To work with these two accounts, configure the AWS CLI with two named profiles. This

example uses profiles named *acctA* and *acctB*, respectively. For information about setting credential profiles and using named profiles, see Configuration and credential file settings in the *AWS Command Line Interface User Guide*.

> **Important**
>
> The profiles that you use for this procedure must have the necessary permissions. For example, in the replication configuration, you specify the IAM role that Amazon S3 can assume. You can do this only if the profile that you use has the `iam:PassRole` permission. If you use administrator user credentials to create a named profile, then you can perform all of the tasks in this procedure. For more information, see Granting a user permissions to pass a role to an AWS service in the *IAM User Guide*.

2. Create the source bucket and enable versioning. This example creates a source bucket named *amzn-s3-demo-source-bucket* in the US East (N. Virginia) (`us-east-1`) Region.

```
aws s3api create-bucket \
--bucket amzn-s3-demo-source-bucket \
--region us-east-1 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket amzn-s3-demo-source-bucket \
--versioning-configuration Status=Enabled \
--profile acctA
```

3. Create a destination bucket and enable versioning. This example creates a destination bucket named *amzn-s3-demo-destination-bucket* in the US West (Oregon) (`us-west-2`) Region. Use an AWS account profile that's different from the one that you used for the source bucket.

```
aws s3api create-bucket \
--bucket amzn-s3-demo-destination-bucket \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctB
```

```
aws s3api put-bucket-versioning \
--bucket amzn-s3-demo-destination-bucket \
```

```
--versioning-configuration Status=Enabled \
--profile acctB
```

4. You must add permissions to your destination bucket policy to allow changing the replica ownership.

   a. Save the following policy to a file named *destination-bucket-policy*.json. Make sure to replace the *user input placeholders* with your own information.

      JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "destination_bucket_policy_sid",
            "Principal": {
                "AWS": "source-bucket-owner-123456789012"
            },
            "Action": [
                "s3:ReplicateObject",
                "s3:ReplicateDelete",
                "s3:ObjectOwnerOverrideToBucketOwner",
                "s3:ReplicateTags",
                "s3:GetObjectVersionTagging"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
            ]
        }
    ]
}
```

   b. Add the preceding policy to the destination bucket by using the following `put-bucket-policy` command:

```
aws s3api put-bucket-policy --region $ {destination-region} --bucket $ {amzn-s3-demo-destination-bucket} --policy file://destination_bucket_policy.json
```

5. Create an IAM role. You specify this role in the replication configuration that you add to the source bucket later. Amazon S3 assumes this role to replicate objects on your behalf. You create an IAM role in two steps:

   - Create the role.
   - Attach a permissions policy to the role.

   a. Create the IAM role.

      i. Copy the following trust policy and save it to a file named *s3-role-trust-policy*.json in the current directory on your local computer. This policy grants Amazon S3 permissions to assume the role.

      JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

      ii. Run the following AWS CLI `create-role` command to create the IAM role:

```
$ aws iam create-role \
--role-name replicationRole \
--assume-role-policy-document file://s3-role-trust-policy.json  \
--profile acctA
```

      Make note of the Amazon Resource Name (ARN) of the IAM role that you created. You will need this ARN in a later step.

   b. Attach a permissions policy to the role.

i. Copy the following permissions policy and save it to a file named *s3-role-perm-pol-changeowner*.json in the current directory on your local computer. This policy grants permissions for various Amazon S3 bucket and object actions. In the following steps, you attach this policy to the IAM role that you created earlier.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ObjectOwnerOverrideToBucketOwner",
            "s3:ReplicateTags",
            "s3:GetObjectVersionTagging"
         ],
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      }
```

```
    ]
}
```

ii. To attach the preceding permissions policy to the role, run the following `put-role-policy` command:

```
$ aws iam put-role-policy \
--role-name replicationRole \
--policy-document file://s3-role-perm-pol-changeowner.json \
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

6. Add a replication configuration to your source bucket.

a. The AWS CLI requires specifying the replication configuration as JSON. Save the following JSON in a file named *`replication`*`.json` in the current directory on your local computer. In the configuration, the `AccessControlTranslation` specifies the change in replica ownership from the source bucket owner to the destination bucket owner.

```
{
   "Role":"IAM-role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
         "Priority":1,
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
         },
         "Status":"Enabled",
         "Destination":{
            "Bucket":"arn:aws:s3:::amzn-s3-demo-destination-bucket",
            "Account":"destination-bucket-owner-account-id",
            "AccessControlTranslation":{
               "Owner":"Destination"
            }
         }
      }
   ]
}
```

b. Edit the JSON by providing values for the destination bucket name, the destination bucket owner account ID, and the *IAM-role-ARN*. Replace *IAM-role-ARN* with the ARN of the IAM role that you created earlier. Save the changes.

c. To add the replication configuration to the source bucket, run the following command:

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket amzn-s3-demo-source-bucket \
--profile acctA
```

7. Test your replication configuration by checking replica ownership in the Amazon S3 console.

a. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

b. Add objects to the source bucket. Verify that the destination bucket contains the object replicas and that the ownership of the replicas has changed to the AWS account that owns the destination bucket.

Using the AWS SDKs

For a code example to add a replication configuration, see Using the AWS SDKs. You must modify the replication configuration appropriately. For conceptual information, see Changing the replica owner.

Meeting compliance requirements with S3 Replication Time Control

S3 Replication Time Control (S3 RTC) helps you meet compliance or business requirements for data replication and provides visibility into Amazon S3 replication times. S3 RTC replicates most objects that you upload to Amazon S3 in seconds, and 99.9 percent of those objects within 15 minutes.

By default, S3 RTC includes two ways to track the progress of replication:

- **S3 Replication metrics** – You can use S3 Replication metrics to monitor the total number of S3 API operations that are pending replication, the total size of objects pending replication, the maximum replication time to the destination Region, and the total number of operations that failed replication. You can then monitor each dataset that you replicate separately. You can also enable S3 Replication metrics independently of S3 RTC. For more information, see the section called “Using S3 Replication metrics”.

Replication rules with S3 Replication Time Control (S3 RTC) enabled publish S3 Replication metrics. Replication metrics are available within 15 minutes of enabling S3 RTC. Replication metrics are available through the Amazon S3 console, the Amazon S3 API, the AWS SDKs, the AWS Command Line Interface (AWS CLI), and Amazon CloudWatch. For more information about CloudWatch metrics, see Monitoring metrics with Amazon CloudWatch. For more information about viewing replication metrics through the Amazon S3 console, see Viewing replication metrics.

S3 Replication metrics are billed at the same rate as Amazon CloudWatch custom metrics. For information, see Amazon CloudWatch pricing.

- **Amazon S3 Event Notifications** – S3 RTC provides `OperationMissedThreshold` and `OperationReplicatedAfterThreshold` events that notify the bucket owner if object replication exceeds or occurs after the 15-minute threshold. With S3 RTC, Amazon S3 Event Notifications can notify you in the rare instance when objects don't replicate within 15 minutes and when those objects replicate after the 15-minute threshold.

  Replication events are available within 15 minutes of enabling S3 RTC. Amazon S3 Event Notifications are available through Amazon SQS, Amazon SNS, or AWS Lambda. For more information, see the section called “Receiving replication failure events”.

Best practices and guidelines for S3 RTC

When replicating data in Amazon S3 with S3 Replication Time Control (S3 RTC) enabled, follow these best practice guidelines to optimize replication performance for your workloads.

Topics

- Amazon S3 Replication and request rate performance guidelines
- Estimating your replication request rates
- Exceeding S3 RTC data transfer rate quotas
- AWS KMS encrypted object replication request rates

Amazon S3 Replication and request rate performance guidelines

When uploading and retrieving storage from Amazon S3, your applications can achieve thousands of transactions per second in request performance. For example, an application can achieve at least

3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per prefix in an S3 bucket, including the requests that S3 Replication makes on your behalf. There are no limits to the number of prefixes in a bucket. You can increase your read or write performance by parallelizing reads. For example, if you create 10 prefixes in an S3 bucket to parallelize reads, you can scale your read performance to 55,000 read requests per second.

Amazon S3 automatically scales in response to sustained request rates above these guidelines, or sustained request rates concurrent with LIST requests. While Amazon S3 is internally optimizing for the new request rate, you might receive HTTP 503 request responses temporarily until the optimization is complete. This behavior might occur with increases in request per second rates, or when you first enable S3 RTC. During these periods, your replication latency might increase. The S3 RTC service level agreement (SLA) doesn't apply to time periods when Amazon S3 performance guidelines on requests per second are exceeded.

The S3 RTC SLA also doesn't apply during time periods where your replication data transfer rate exceeds the default 1 gigabit per second (Gbps) quota. If you expect your replication transfer rate to exceed 1 Gbps, you can contact AWS Support Center or use Service Quotas to request an increase in your replication transfer rate quota.

Estimating your replication request rates

Your total request rate including the requests that Amazon S3 replication makes on your behalf must be within the Amazon S3 request rate guidelines for both the replication source and destination buckets. For each object replicated, Amazon S3 replication makes up to five GET/HEAD requests and one PUT request to the source bucket, and one PUT request to each destination bucket.

For example, if you expect to replicate 100 objects per second, Amazon S3 replication might perform an additional 100 PUT requests on your behalf for a total of 200 PUT requests per second to the source S3 bucket. Amazon S3 replication also might perform up to 500 GET/HEAD requests (5 GET/HEAD requests for each object that's replicated.)

> **Note**
>
> You incur costs for only one PUT request per object replicated. For more information, see the pricing information in the Amazon S3 FAQs about replication.

Exceeding S3 RTC data transfer rate quotas

If you expect your S3 RTC data transfer rate to exceed the default 1 Gbps quota, contact AWS Support Center or use Service Quotas to request an increase in your replication transfer rate quota. When calculating your transfer rate, include annotation data because annotation bytes count toward the S3 RTC data transfer rate quota.

AWS KMS encrypted object replication request rates

When you replicate objects that are encrypted with server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), AWS KMS requests per second quotas apply. AWS KMS might reject an otherwise valid request because your request rate exceeds the quota for the number of requests per second. When a request is throttled, AWS KMS returns a `ThrottlingException` error. The AWS KMS request rate quota applies to requests that you make directly and to requests made by Amazon S3 replication on your behalf.

For example, if you expect to replicate 1,000 objects per second, you can subtract 2,000 requests from your AWS KMS request rate quota. The resulting request rate per second is available for your AWS KMS workloads excluding replication. You can use AWS KMS request metrics in Amazon CloudWatch to monitor the total AWS KMS request rate on your AWS account.

To request an increase to your AWS KMS requests per second quota, contact AWS Support Center or use Service Quotas.

Enabling S3 Replication Time Control

You can start using S3 Replication Time Control (S3 RTC) with a new or existing replication rule. You can choose to apply your replication rule to an entire bucket, or to objects with a specific prefix or tag. When you enable S3 RTC, S3 Replication metrics are also enabled on your replication rule.

You can configure S3 RTC by using the Amazon S3 console, the Amazon S3 API, the AWS SDKs, and the AWS Command Line Interface (AWS CLI).

Using the S3 console

For step-by-step instructions, see Configuring replication for buckets in the same account. This topic provides instructions for enabling S3 RTC in your replication configuration when the source and destination buckets are owned by the same and different AWS accounts.

Using the AWS CLI

To use the AWS CLI to replicate objects with S3 RTC enabled, you create buckets, enable versioning on the buckets, create an IAM role that gives Amazon S3 permission to replicate objects, and add the replication configuration to the source bucket. The replication configuration must have S3 RTC enabled, as shown in the following example.

For step-by-step instructions for setting up your replication configuration by using the AWS CLI, see Configuring replication for buckets in the same account.

The following example replication configuration enables and sets the `ReplicationTime` and `EventThreshold` values for a replication rule. Enabling and setting these values enables S3 RTC on the rule.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Disabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::amzn-s3-demo-destination-bucket",
                "Metrics": {
                    "Status": "Enabled",
                    "EventThreshold": {
                        "Minutes": 15
                    }
                },
                "ReplicationTime": {
                    "Status": "Enabled",
                    "Time": {
                        "Minutes": 15
                    }
                }
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
```

```
}
```

> **Important**
>
> `Metrics:EventThreshold:Minutes` and `ReplicationTime:Time:Minutes` can only have 15 as a valid value.

Using the AWS SDK for Java

The following Java example adds replication configuration with S3 Replication Time Control (S3 RTC) enabled.

```
import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.DeleteMarkerReplication;
import software.amazon.awssdk.services.s3.model.Destination;
import software.amazon.awssdk.services.s3.model.Metrics;
import software.amazon.awssdk.services.s3.model.MetricsStatus;
import software.amazon.awssdk.services.s3.model.PutBucketReplicationRequest;
import software.amazon.awssdk.services.s3.model.ReplicationConfiguration;
import software.amazon.awssdk.services.s3.model.ReplicationRule;
import software.amazon.awssdk.services.s3.model.ReplicationRuleFilter;
import software.amazon.awssdk.services.s3.model.ReplicationTime;
import software.amazon.awssdk.services.s3.model.ReplicationTimeStatus;
import software.amazon.awssdk.services.s3.model.ReplicationTimeValue;

public class Main {

  public static void main(String[] args) {
    S3Client s3 = S3Client.builder()
      .region(Region.US_EAST_1)
      .credentialsProvider(() -> AwsBasicCredentials.create(
          "AWS_ACCESS_KEY_ID",
          "AWS_SECRET_ACCESS_KEY")
      )
      .build();

    ReplicationConfiguration replicationConfig = ReplicationConfiguration
      .builder()
      .rules(
          ReplicationRule
```

```
            .builder()
            .status("Enabled")
            .priority(1)
            .deleteMarkerReplication(
                DeleteMarkerReplication
                    .builder()
                    .status("Disabled")
                    .build()
            )
            .destination(
                Destination
                    .builder()
                    .bucket("destination_bucket_arn")
                    .replicationTime(
                        ReplicationTime.builder().time(
                            ReplicationTimeValue.builder().minutes(15).build()
                        ).status(
                            ReplicationTimeStatus.ENABLED
                        ).build()
                    )
                    .metrics(
                        Metrics.builder().eventThreshold(
                            ReplicationTimeValue.builder().minutes(15).build()
                        ).status(
                            MetricsStatus.ENABLED
                        ).build()
                    )
                    .build()
            )
            .filter(
                ReplicationRuleFilter
                    .builder()
                    .prefix("testtest")
                    .build()
            )
        .build())
        .role("role_arn")
        .build();

    // Put replication configuration
    PutBucketReplicationRequest putBucketReplicationRequest =
PutBucketReplicationRequest
      .builder()
      .bucket("source_bucket")
```

```
      .replicationConfiguration(replicationConfig)
      .build();

    s3.putBucketReplication(putBucketReplicationRequest);
  }
}
```

Replicating encrypted objects (SSE-S3, SSE-KMS, DSSE-KMS, SSE-C)

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

There are some special considerations when you're replicating objects that have been encrypted by using server-side encryption. Amazon S3 supports the following types of server-side encryption:

- Server-side encryption with Amazon S3 managed keys (SSE-S3)
- Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS)
- Dual-layer server-side encryption with AWS KMS keys (DSSE-KMS)
- Server-side encryption with customer-provided keys (SSE-C)

For more information about server-side encryption, see the section called “Server-side encryption”.

This topic explains the permissions that you need to direct Amazon S3 to replicate objects that have been encrypted by using server-side encryption. This topic also provides additional configuration elements that you can add and example AWS Identity and Access Management (IAM) policies that grant the necessary permissions for replicating encrypted objects.

For an example with step-by-step instructions, see Enabling replication for encrypted objects. For information about creating a replication configuration, see Replicating objects within and across Regions.

> **Note**
>
> You can use multi-Region AWS KMS keys in Amazon S3. However, Amazon S3 currently treats multi-Region keys as though they were single-Region keys, and does not use the multi-Region features of the key. For more information, see Using multi-Region keys in the *AWS Key Management Service Developer Guide*.

Topics

- How default bucket encryption affects replication
- Replicating objects encrypted with SSE-C
- Replicating objects encrypted with SSE-S3, SSE-KMS, or DSSE-KMS
- Enabling replication for encrypted objects

How default bucket encryption affects replication

When you enable default encryption for a replication destination bucket, the following encryption behavior applies:

- If objects in the source bucket are not encrypted, the replica objects in the destination bucket are encrypted by using the default encryption settings of the destination bucket. As a result, the entity tags (ETags) of the source objects differ from the ETags of the replica objects. If you have applications that use ETags, you must update those applications to account for this difference.
- If objects in the source bucket are encrypted by using server-side encryption with Amazon S3 managed keys (SSE-S3), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), the replica objects in the destination bucket use the same type of encryption as the source objects. The default encryption settings of the destination bucket are not used.

Replicating objects encrypted with SSE-C

By using server-side encryption with customer-provided keys (SSE-C), you can manage your own proprietary encryption keys. With SSE-C, you manage the keys while Amazon S3 manages the

encryption and decryption process. You must provide an encryption key as part of your request, but you don't need to write any code to perform object encryption or decryption. When you upload an object, Amazon S3 encrypts the object by using the key that you provided. Amazon S3 then purges that key from memory. When you retrieve an object, you must provide the same encryption key as part of your request. For more information, see the section called "Customer-provided encryption keys (SSE-C)".

S3 Replication supports objects that are encrypted with SSE-C. You can configure SSE-C object replication in the Amazon S3 console or with the AWS SDKs in the same way that you configure replication for unencrypted objects. There aren't additional SSE-C permissions beyond what are currently required for replication.

S3 Replication automatically replicates newly uploaded SSE-C encrypted objects if they are eligible, as specified in your S3 Replication configuration. To replicate existing objects in your buckets, use S3 Batch Replication. For more information about replicating objects, see the section called "Setting up live replication" and the section called "Replicating existing objects".

There are no additional charges for replicating SSE-C objects. For details about replication pricing, see Amazon S3 pricing.

Replicating objects encrypted with SSE-S3, SSE-KMS, or DSSE-KMS

By default, Amazon S3 doesn't replicate objects that are encrypted with SSE-KMS or DSSE-KMS. This section explains the additional configuration elements that you can add to direct Amazon S3 to replicate these objects.

For an example with step-by-step instructions, see Enabling replication for encrypted objects. For information about creating a replication configuration, see Replicating objects within and across Regions.

Specifying additional information in the replication configuration

In the replication configuration, you do the following:

- In the `Destination` element in your replication configuration, add the ID of the symmetric AWS KMS customer managed key that you want Amazon S3 to use to encrypt object replicas, as shown in the following example replication configuration.
- Explicitly opt in by enabling replication of objects encrypted by using KMS keys (SSE-KMS or DSSE-KMS). To opt in, add the `SourceSelectionCriteria` element, as shown in the following example replication configuration.

```
<ReplicationConfiguration>
   <Rule>
      ...
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
           <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>

      <Destination>
          ...
          <EncryptionConfiguration>
             <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same AWS
 Region as the destination bucket.</ReplicaKmsKeyID>
          </EncryptionConfiguration>
       </Destination>
      ...
   </Rule>
</ReplicationConfiguration>
```

Important

- The KMS key must have been created in the same AWS Region as the destination bucket.
- The KMS key *must* be valid. The `PutBucketReplication` API operation doesn't check the validity of KMS keys. If you use a KMS key that isn't valid, you will receive the HTTP `200 OK` status code in response, but replication fails.

The following example shows a replication configuration that includes optional configuration elements. This replication configuration has one rule. The rule applies to objects with the *Tax* key prefix. Amazon S3 uses the specified AWS KMS key ID to encrypt these object replicas.

```
<?xml version="1.0" encoding="UTF-8"?>
<ReplicationConfiguration>
   <Role>arn:aws:iam::account-id:role/role-name</Role>
   <Rule>
      <ID>Rule-1</ID>
      <Priority>1</Priority>
      <Status>Enabled</Status>
      <DeleteMarkerReplication>
```

```
         <Status>Disabled</Status>
      </DeleteMarkerReplication>
      <Filter>
         <Prefix>Tax</Prefix>
      </Filter>
      <Destination>
         <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
         <EncryptionConfiguration>
            <ReplicaKmsKeyID>AWS KMS key ARN or Key Alias ARN that's in the same AWS
 Region as the destination bucket.</ReplicaKmsKeyID>
         </EncryptionConfiguration>
      </Destination>
      <SourceSelectionCriteria>
         <SseKmsEncryptedObjects>
            <Status>Enabled</Status>
         </SseKmsEncryptedObjects>
      </SourceSelectionCriteria>
   </Rule>
</ReplicationConfiguration>
```

Granting additional permissions for the IAM role

To replicate objects that are encrypted at rest by using SSE-S3, SSE-KMS, or DSSE-KMS, grant the following additional permissions to the AWS Identity and Access Management (IAM) role that you specify in the replication configuration. You grant these permissions by updating the permissions policy that's associated with the IAM role.

- **`s3:GetObjectVersionForReplication` action for source objects** – This action allows Amazon S3 to replicate both unencrypted objects and objects created with server-side encryption by using SSE-S3, SSE-KMS, or DSSE-KMS.

  > **Note**
  >
  > We recommend that you use the `s3:GetObjectVersionForReplication` action instead of the `s3:GetObjectVersion` action because `s3:GetObjectVersionForReplication` provides Amazon S3 with only the minimum permissions necessary for replication. In addition, the `s3:GetObjectVersion` action allows replication of unencrypted and SSE-S3-encrypted objects, but not of objects that are encrypted by using KMS keys (SSE-KMS or DSSE-KMS).

- **`kms:Decrypt` and `kms:Encrypt` AWS KMS actions for the KMS keys**

- You must grant `kms:Decrypt` permissions for the AWS KMS key that protects the source object.
- You must grant `kms:Encrypt` permissions for the AWS KMS key that will protect the object replica.

- **`kms:GenerateDataKey` action for replicating plaintext objects** – If you're replicating plaintext objects to a bucket with SSE-KMS or DSSE-KMS encryption enabled by default, you must include the `kms:GenerateDataKey` permission for the destination encryption context and the KMS key in the IAM policy.

> **Important**
>
> If you use S3 Batch Replication to replicate datasets cross region and your objects previously had their server-side encryption type updated from SSE-S3 to SSE-KMS, you may need additional permissions. On the source region bucket, you must have `kms:decrypt` permissions. Then, you will need the `kms:decrypt` and `kms:encrypt` permissions for the bucket in the destination region.

We recommend that you restrict these permissions only to the destination buckets and objects by using AWS KMS condition keys. The AWS account that owns the IAM role must have permissions for the `kms:Encrypt` and `kms:Decrypt` actions for the KMS keys that are listed in the policy. If the KMS keys are owned by another AWS account, the owner of the KMS keys must grant these permissions to the AWS account that owns the IAM role. For more information about managing access to these KMS keys, see Using IAM policies with AWS KMS in the *AWS Key Management Service Developer Guide*.

S3 Bucket Keys and replication

To use replication with an S3 Bucket Key, the AWS KMS key policy for the KMS key that's used to encrypt the object replica must include the `kms:Decrypt` permission for the calling principal. The call to `kms:Decrypt` verifies the integrity of the S3 Bucket Key before using it. For more information, see Using an S3 Bucket Key with replication.

When an S3 Bucket Key is enabled for the source or destination bucket, the encryption context will be the bucket's Amazon Resource Name (ARN), not the object's ARN (for example, `arn:aws:s3:::`*`bucket_ARN`*). You must update your IAM policies to use the bucket ARN for the encryption context:

```
"kms:EncryptionContext:aws:s3:arn": [
"arn:aws:s3:::bucket_ARN"
]
```

For more information, see Encryption context (`x-amz-server-side-encryption-context`) (in the "Using the REST API" section) and Changes to note before enabling an S3 Bucket Key.

Example policies: Using SSE-S3 and SSE-KMS with replication

The following example IAM policies show statements for using SSE-S3 and SSE-KMS with replication.

Example– Using SSE-KMS with separate destination buckets

The following example policy shows statements for using SSE-KMS with separate destination buckets.

Example– Replicating objects created with SSE-S3 and SSE-KMS

The following is a complete IAM policy that grants the necessary permissions to replicate unencrypted objects, objects created with SSE-S3, and objects created with SSE-KMS.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
```

```
            ],
            "Resource":[
               "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
            ]
         },
         {
            "Effect":"Allow",
            "Action":[
               "s3:ReplicateObject",
               "s3:ReplicateDelete"
            ],
            "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/key-prefix1*"
         },
         {
            "Action":[
               "kms:Decrypt"
            ],
            "Effect":"Allow",
            "Condition":{
               "StringLike":{
                  "kms:ViaService":"s3.us-east-1.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn":[
                     "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
                  ]
               }
            },
            "Resource":[
              "arn:aws:kms:us-east-1:111122223333:key/key-id"
            ]
         },
         {
            "Action":[
               "kms:Encrypt"
            ],
            "Effect":"Allow",
            "Condition":{
               "StringLike":{
                  "kms:ViaService":"s3.us-east-1.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn":[
                     "arn:aws:s3:::amzn-s3-demo-destination-bucket/prefix1*"
                  ]
               }
            },
            "Resource":[
```

```
            "arn:aws:kms:us-east-1:111122223333:key/key-id"
         ]
      }
   ]
}
```

Example– Replicating objects with S3 Bucket Keys

The following is a complete IAM policy that grants the necessary permissions to replicate objects with S3 Bucket Keys.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetReplicationConfiguration",
            "s3:ListBucket"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl"
         ],
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket/key-prefix1*"
         ]
      },
      {
         "Effect":"Allow",
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete"
         ],
```

```
            "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/key-prefix1*"
         },
         {
            "Action":[
               "kms:Decrypt"
            ],
            "Effect":"Allow",
            "Condition":{
               "StringLike":{
                  "kms:ViaService":"s3.us-east-1.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn":[
                     "arn:aws:s3:::amzn-s3-demo-source-bucket"
                  ]
               }
            },
            "Resource":[
              "arn:aws:kms:us-east-1:111122223333:key/key-id"
            ]
         },
         {
            "Action":[
               "kms:Encrypt"
            ],
            "Effect":"Allow",
            "Condition":{
               "StringLike":{
                  "kms:ViaService":"s3.us-east-1.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn":[
                     "arn:aws:s3:::amzn-s3-demo-destination-bucket"
                  ]
               }
            },
            "Resource":[
               "arn:aws:kms:us-east-1:111122223333:key/key-id"
            ]
         }
      ]
   }
```

Granting additional permissions for cross-account scenarios

In a cross-account scenario, where the source and destination buckets are owned by different AWS accounts, you can use a KMS key to encrypt object replicas. However, the KMS key owner must grant the source bucket owner permission to use the KMS key.

> **Note**
>
> If you need to replicate SSE-KMS data cross-account, then your replication rule must specify a customer managed key from AWS KMS for the destination account. AWS managed keys don't allow cross-account use, and therefore can't be used to perform cross-account replication.

To grant the source bucket owner permission to use the KMS key (AWS KMS console)

1. Sign in to the AWS Management Console and open the AWS KMS console at https://console.aws.amazon.com/kms.
2. To change the AWS Region, use the Region selector in the upper-right corner of the page.
3. To view the keys in your account that you create and manage, in the navigation pane choose **Customer managed keys**.
4. Choose the KMS key.
5. Under the **General configuration** section, choose the **Key policy** tab.
6. Scroll down to **Other AWS accounts**.
7. Choose **Add other AWS accounts**.

   The **Other AWS accounts** dialog box appears.
8. In the dialog box, choose **Add another AWS account**. For **arn:aws:iam::**, enter the source bucket account ID.
9. Choose **Save changes**.

To grant the source bucket owner permission to use the KMS key (AWS CLI)

- For information about the `put-key-policy` AWS Command Line Interface (AWS CLI) command, see put-key-policy in the *AWS CLI Command Reference*. For information about the underlying `PutKeyPolicy` API operation, see PutKeyPolicy in the AWS Key Management Service API Reference.

AWS KMS transaction quota considerations

When you add many new objects with AWS KMS encryption after enabling Cross-Region Replication (CRR), you might experience throttling (HTTP `503 Service Unavailable` errors). Throttling occurs when the number of AWS KMS transactions per second exceeds the current quota. For more information, see Quotas in the *AWS Key Management Service Developer Guide*.

To request a quota increase, use Service Quotas. For more information, see Requesting a quota increase. If Service Quotas isn't supported in your Region, open an AWS Support case.

Enabling replication for encrypted objects

By default, Amazon S3 doesn't replicate objects that are encrypted by using server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) or dual-layer server-side encryption with AWS KMS keys (DSSE-KMS). To replicate objects encrypted with SSE-KMS or DSS-KMS, you must modify the bucket replication configuration to tell Amazon S3 to replicate these objects. This example explains how to use the Amazon S3 console and the AWS Command Line Interface (AWS CLI) to change the bucket replication configuration to enable replicating encrypted objects.

Note

When an S3 Bucket Key is enabled for the source or destination bucket, the encryption context will be the bucket's Amazon Resource Name (ARN), not the object's ARN. You must update your IAM policies to use the bucket ARN for the encryption context. For more information, see S3 Bucket Keys and replication.

Note

You can use multi-Region AWS KMS keys in Amazon S3. However, Amazon S3 currently treats multi-Region keys as though they were single-Region keys, and does not use the multi-Region features of the key. For more information, see Using multi-Region keys in the *AWS Key Management Service Developer Guide*.

Using the S3 console

For step-by-step instructions, see Configuring replication for buckets in the same account. This topic provides instructions for setting a replication configuration when the source and destination buckets are owned by the same and different AWS accounts.

Using the AWS CLI

To replicate encrypted objects with the AWS CLI, you do the following:

- Create source and destination buckets and enable versioning on these buckets.
- Create an AWS Identity and Access Management (IAM) service role that gives Amazon S3 permission to replicate objects. The IAM role's permissions include the necessary permissions to replicate the encrypted objects.
- Add a replication configuration to the source bucket. The replication configuration provides information related to replicating objects that are encrypted by using KMS keys.
- Add encrypted objects to the source bucket.
- Test the setup to confirm that your encrypted objects are being replicated to the destination bucket.

The following procedures walk you through this process.

To replicate server-side encrypted objects (AWS CLI)

To use the examples in this procedure, replace the *`user input placeholders`* with your own information.

1. In this example, you create both the source (*`amzn-s3-demo-source-bucket`*) and destination (*`amzn-s3-demo-destination-bucket`*) buckets in the same AWS account. You also set a credentials profile for the AWS CLI. This example uses the profile name *`acctA`*.

   For information about setting credential profiles and using named profiles, see Configuration and credential file settings in the *AWS Command Line Interface User Guide*.
2. Use the following commands to create the *`amzn-s3-demo-source-bucket`* bucket and enable versioning on it. The following example commands create the *`amzn-s3-demo-source-bucket`* bucket in the US East (N. Virginia) (`us-east-1`) Region.

   ```
   aws s3api create-bucket \
   --bucket amzn-s3-demo-source-bucket \
   --region us-east-1 \
   --profile acctA
   ```

   ```
   aws s3api put-bucket-versioning \
   --bucket amzn-s3-demo-source-bucket \
   ```

```
--versioning-configuration Status=Enabled \
--profile acctA
```

3. Use the following commands to create the *amzn-s3-demo-destination-bucket* bucket and enable versioning on it. The following example commands create the *amzn-s3-demo-destination-bucket* bucket in the US West (Oregon) (`us-west-2`) Region.

> **Note**
>
> To set up a replication configuration when both *amzn-s3-demo-source-bucket* and *amzn-s3-demo-destination-bucket* buckets are in the same AWS account, you use the same profile. This example uses *acctA*. To configure replication when the buckets are owned by different AWS accounts, you specify different profiles for each.

```
aws s3api create-bucket \
--bucket amzn-s3-demo-destination-bucket \
--region us-west-2 \
--create-bucket-configuration LocationConstraint=us-west-2 \
--profile acctA
```

```
aws s3api put-bucket-versioning \
--bucket amzn-s3-demo-destination-bucket \
--versioning-configuration Status=Enabled \
--profile acctA
```

4. Next, you create an IAM service role. You will specify this role in the replication configuration that you add to the *amzn-s3-demo-source-bucket* bucket later. Amazon S3 assumes this role to replicate objects on your behalf. You create an IAM role in two steps:

   - Create a service role.
   - Attach a permissions policy to the role.

   a. To create an IAM service role, do the following:

      i. Copy the following trust policy and save it to a file called *s3-role-trust-policy-kmsobj*`.json` in the current directory on your local computer. This policy grants the

Amazon S3 service principal permissions to assume the role so that Amazon S3 can perform tasks on your behalf.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

ii. Use the following command to create the role:

```
$ aws iam create-role \
--role-name replicationRolekmsobj \
--assume-role-policy-document file://s3-role-trust-policy-kmsobj.json  \
--profile acctA
```

b. Next, you attach a permissions policy to the role. This policy grants permissions for various Amazon S3 bucket and object actions.

i. Copy the following permissions policy and save it to a file named *s3-role-permissions-policykmsobj*.json in the current directory on your local computer. You will create an IAM role and attach the policy to it later.

> **Important**
>
> In the permissions policy, you specify the AWS KMS key IDs that will be used for encryption of the *amzn-s3-demo-source-bucket* and *amzn-s3-demo-destination-bucket* buckets. You must create two separate KMS keys for the *amzn-s3-demo-source-bucket* and *amzn-s3-demo-destination-*

bucket buckets. AWS KMS keys aren't shared outside the AWS Region in which they were created.

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Action":[
            "s3:ListBucket",
            "s3:GetReplicationConfiguration",
            "s3:GetObjectVersionForReplication",
            "s3:GetObjectVersionAcl",
            "s3:GetObjectVersionTagging"
         ],
         "Effect":"Allow",
         "Resource":[
            "arn:aws:s3:::amzn-s3-demo-source-bucket",
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Action":[
            "s3:ReplicateObject",
            "s3:ReplicateDelete",
            "s3:ReplicateTags"
         ],
         "Effect":"Allow",
         "Condition":{
            "StringLikeIfExists":{
               "s3:x-amz-server-side-encryption":[
                  "aws:kms",
                  "AES256",
                  "aws:kms:dsse"
               ],
               "s3:x-amz-server-side-encryption-aws-kms-key-id":[
                  "AWS KMS key IDs(in ARN format) to use for
 encrypting object replicas"
               ]
            }
```

```
            },
            "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
         },
         {
            "Action":[
               "kms:Decrypt"
            ],
            "Effect":"Allow",
            "Condition":{
               "StringLike":{
                  "kms:ViaService":"s3.us-east-1.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn":[
                     "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
                  ]
               }
            },
            "Resource":[
               "arn:aws:kms:us-east-1:111122223333:key/key-id"
            ]
         },
         {
            "Action":[
               "kms:Encrypt"
            ],
            "Effect":"Allow",
            "Condition":{
               "StringLike":{
                  "kms:ViaService":"s3.us-west-2.amazonaws.com",
                  "kms:EncryptionContext:aws:s3:arn":[
                     "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
                  ]
               }
            },
            "Resource":[
               "arn:aws:kms:us-west-2:111122223333:key/key-id"
            ]
         }
      ]
}
```

ii. Create a policy and attach it to the role.

```
$ aws iam put-role-policy \
--role-name replicationRolekmsobj \
```

```
--policy-document file://s3-role-permissions-policykmsobj.json \
--policy-name replicationRolechangeownerPolicy \
--profile acctA
```

5. Next, add the following replication configuration to the *amzn-s3-demo-source-bucket* bucket. It tells Amazon S3 to replicate objects with the *Tax/* prefix to the *amzn-s3-demo-destination-bucket* bucket.

> **Important**
>
> In the replication configuration, you specify the IAM role that Amazon S3 can assume. You can do this only if you have the `iam:PassRole` permission. The profile that you specify in the CLI command must have this permission. For more information, see Granting a user permissions to pass a role to an AWS service in the *IAM User Guide*.

```
<ReplicationConfiguration>
 <Role>IAM-Role-ARN</Role>
 <Rule>
   <Priority>1</Priority>
   <DeleteMarkerReplication>
      <Status>Disabled</Status>
   </DeleteMarkerReplication>
   <Filter>
      <Prefix>Tax</Prefix>
   </Filter>
   <Status>Enabled</Status>
   <SourceSelectionCriteria>
     <SseKmsEncryptedObjects>
       <Status>Enabled</Status>
     </SseKmsEncryptedObjects>
   </SourceSelectionCriteria>
   <Destination>
     <Bucket>arn:aws:s3:::amzn-s3-demo-destination-bucket</Bucket>
     <EncryptionConfiguration>
       <ReplicaKmsKeyID>AWS KMS key IDs to use for encrypting object replicas</
ReplicaKmsKeyID>
     </EncryptionConfiguration>
   </Destination>
  </Rule>
</ReplicationConfiguration>
```

To add a replication configuration to the *amzn-s3-demo-source-bucket* bucket, do the following:

a. The AWS CLI requires you to specify the replication configuration as JSON. Save the following JSON in a file (*replication*.json) in the current directory on your local computer.

```
{
   "Role":"IAM-Role-ARN",
   "Rules":[
      {
         "Status":"Enabled",
         "Priority":1,
         "DeleteMarkerReplication":{
            "Status":"Disabled"
         },
         "Filter":{
            "Prefix":"Tax"
         },
         "Destination":{
            "Bucket":"arn:aws:s3:::amzn-s3-demo-destination-bucket",
            "EncryptionConfiguration":{
               "ReplicaKmsKeyID":"AWS KMS key IDs (in ARN format) to use for
 encrypting object replicas"
            }
         },
         "SourceSelectionCriteria":{
            "SseKmsEncryptedObjects":{
               "Status":"Enabled"
            }
         }
      }
   ]
}
```

b. Edit the JSON to provide values for the *amzn-s3-demo-destination-bucket* bucket, *AWS KMS key IDs (in ARN format)*, and *IAM-role-ARN*. Save the changes.

c. Use the following command to add the replication configuration to your *amzn-s3-demo-source-bucket* bucket. Be sure to provide the *amzn-s3-demo-source-bucket* bucket name.

```
$ aws s3api put-bucket-replication \
--replication-configuration file://replication.json \
--bucket amzn-s3-demo-source-bucket \
--profile acctA
```

6. Test the configuration to verify that encrypted objects are replicated. In the Amazon S3 console, do the following:

   a. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

   b. In the *amzn-s3-demo-source-bucket* bucket, create a folder named *Tax*.

   c. Add sample objects to the folder. Be sure to choose the encryption option and specify your KMS key to encrypt the objects.

   d. Verify that the *amzn-s3-demo-destination-bucket* bucket contains the object replicas and that they are encrypted by using the KMS key that you specified in the configuration. For more information, see the section called "Getting replication status".

Using the AWS SDKs

For a code example that shows how to add a replication configuration, see Using the AWS SDKs. You must modify the replication configuration appropriately.

Replicating metadata changes with replica modification sync

Amazon S3 replica modification sync can help you keep object metadata such as tags, access control lists (ACLs), annotations, and Object Lock settings replicated between replicas and source objects. By default, Amazon S3 replicates metadata from the source objects to the replicas only. When replica modification sync is enabled, Amazon S3 replicates metadata changes made to the replica copies back to the source object, making the replication bidirectional (two-way replication).

Enabling replica modification sync

You can use Amazon S3 replica modification sync with new or existing replication rules. You can apply it to an entire bucket or to objects that have a specific prefix.

To enable replica modification sync by using the Amazon S3 console, see Examples for configuring live replication. This topic provides instructions for enabling replica modification sync in your

replication configuration when the source and destination buckets are owned by the same or different AWS accounts.

To enable replica modification sync by using the AWS Command Line Interface (AWS CLI), you must add a replication configuration to the bucket containing the replicas with `ReplicaModifications` enabled. To set up two-way replication, create a replication rule from the source bucket (*amzn-s3-demo-source-bucket*) to the bucket containing the replicas (*amzn-s3-demo-destination-bucket*). Then, create a second replication rule from the bucket containing the replicas (*amzn-s3-demo-destination-bucket*) to the source bucket (*amzn-s3-demo-source-bucket*). The source and destination buckets can be in the same or different AWS Regions.

> **Note**
>
> You must enable replica modification sync on both the source and destination buckets to replicate replica metadata changes like object access control lists (ACLs), object tags, object annotations, or Object Lock settings on the replicated objects. Like all replication rules, you can apply these rules to the entire bucket or to a subset of objects filtered by prefix or object tags.

In the following example configuration, Amazon S3 replicates metadata changes under the prefix *Tax* to the bucket *amzn-s3-demo-source-bucket*, which contains the source objects.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "SourceSelectionCriteria": {
                "ReplicaModifications":{
                    "Status": "Enabled"
                }
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::amzn-s3-demo-source-bucket"
            },
            "Priority": 1
```

```
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

For full instructions on creating replication rules by using the AWS CLI, see Configuring replication for buckets in the same account.

Replicating delete markers between buckets

By default, when S3 Replication is enabled and an object is deleted in the source bucket, Amazon S3 adds a delete marker in the source bucket only. This action helps protect data in the destination buckets from accidental or malicious deletions. If you have *delete marker replication* enabled, these markers are copied to the destination buckets, and Amazon S3 behaves as if the object was deleted in both the source and destination buckets. For more information about how delete markers work, see Working with delete markers.

Note

- Delete marker replication isn't supported for tag-based replication rules. Delete marker replication also doesn't adhere to the 15-minute service-level agreement (SLA) that's granted when you're using S3 Replication Time Control (S3 RTC).
- If you're not using the latest replication configuration XML version, delete operations affect replication differently. For more information, see How delete operations affect replication.
- If you enable delete marker replication and your source bucket has an S3 Lifecycle expiration rule, the delete markers added by the S3 Lifecycle expiration rule won't be replicated to the destination bucket.

Enabling delete marker replication

You can start using delete marker replication with a new or existing replication rule. You can apply delete marker replication to an entire bucket or to objects that have a specific prefix.

To enable delete marker replication by using the Amazon S3 console, see Using the S3 console. This topic provides instructions for enabling delete marker replication in your replication configuration when the source and destination buckets are owned by the same or different AWS accounts.

To enable delete marker replication by using the AWS Command Line Interface (AWS CLI), you must add a replication configuration to the source bucket with `DeleteMarkerReplication` enabled, as shown in the following example configuration.

In the following example replication configuration, delete markers are replicated to the destination bucket *amzn-s3-demo-destination-bucket* for objects under the prefix *Tax*.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
            },
            "DeleteMarkerReplication": {
                "Status": "Enabled"
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::amzn-s3-demo-destination-bucket"
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

For full instructions on creating replication rules through the AWS CLI, see Configuring replication for buckets in the same account.

Replicating annotations

When you configure replication on a bucket, Amazon S3 replicates annotations automatically as part of live replication. Each annotation replicates independently as changes occur. To replicate annotations, the replication IAM role must include the `s3:GetObjectVersionAnnotationForReplication` permission on the source bucket.

The following example shows the source bucket statement in the IAM role permissions policy with the annotation replication permission included:

```
{
   "Effect": "Allow",
```

```
   "Action": [
      "s3:GetObjectVersionForReplication",
      "s3:GetObjectVersionAcl",
      "s3:GetObjectVersionTagging",
      "s3:GetObjectVersionAnnotationForReplication"
   ],
   "Resource": [
      "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
   ]
}
```

No additional destination bucket permissions are required for annotation replication. The `s3:ReplicateObject` permission on the destination bucket covers annotation replication.

Note the following behaviors for annotation replication:

- Annotation additions and updates on the source are replicated to the destination. Each annotation replicates independently as changes occur.
- Annotation deletions on the source are not replicated to the destination. If you delete an annotation on the source, the annotation persists on the destination.
- If you repeatedly delete and recreate annotations on the source with different names, annotations accumulate on the destination over time because deletions are not replicated.

If you don't want annotations to replicate to a destination bucket, the destination bucket owner can add a Deny statement for the `s3:ReplicateObjectAnnotation` action to the destination bucket policy. Object replication continues to succeed; only annotation replication is blocked. The following example shows the Deny statement:

```
...
   "Statement":[
      {
         "Effect":"Deny",
         "Principal":{
            "AWS":"arn:aws:iam::source-bucket-account-id:role/service-role/source-
account-IAM-role"
         },
         "Action":"s3:ReplicateObjectAnnotation",
         "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
      }
   ]
```

```
...
```

For more information about annotation replication behavior, see the section called "What's replicated?". For permission setup instructions, see Setting up permissions for live replication.

Managing or pausing live replication

Live replication is the automatic, asynchronous copying of objects across buckets in the same or different AWS Regions. After you set up your replication configuration, Amazon S3 replicates newly created objects and object updates from a source bucket to one or more specified destination buckets.

You use the Amazon S3 console to add replication rules to the source bucket. Replication rules define the source bucket objects to replicate and the destination bucket or buckets where the replicated objects are stored. For more information about replication, see Replicating objects within and across Regions.

You can manage replication rules on the **Replication** page in the Amazon S3 console. You can add, view, edit, enable, disable, or delete replication rules. You can also change the priority of your replication rules. For information about adding replication rules to a bucket, see Using the S3 console.

To manage the replication rules for a bucket by using the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. On the **General purpose buckets** tab, choose the name of the bucket that you want.
4. Choose the **Management** tab, and then scroll down to **Replication rules**.
5. You can change your replication rules in the following ways:
   - To enable or disable a replication rule, choose the option button to the left of the rule. On the **Actions** menu, choose **Enable rule** or **Disable rule**. You can also disable, enable, or delete all the rules in the bucket from the **Actions** menu.

     Note

     If you disable a replication rule and then later re-enable the rule, any new or changed objects that weren't replicated while the rule was disabled are *not*

automatically replicated when the rule is re-enabled. To replicate those objects, you must use S3 Batch Replication. For more information, see the section called "Replicating existing objects".

- To change the priority of a rule, choose the option button to the left of the rule, and then choose **Edit rule**.

  You set rule priorities to avoid conflicts caused by objects that are included in the scope of more than one rule. In the case of overlapping rules, Amazon S3 uses the rule priority to determine which rule to apply. The higher the number, the higher the priority. For more information about rule priority, see Replication configuration file elements.

Pausing or stopping replication

To temporarily pause replication and have it automatically resume later, you can use the `aws:s3:bucket-pause-replication` action in AWS Fault Injection Service. For more information, see aws:s3:bucket-pause-replication and Pause S3 Replication in the *AWS Fault Injection Service User Guide*.

To stop replication in Amazon S3, we recommend disabling your replication rules. If you disable a replication rule and then later re-enable the rule, any new or changed objects that weren't replicated while the rule was disabled are *not* automatically replicated when the rule is re-enabled. To replicate those objects, you must use S3 Batch Replication. For more information, see the section called "Replicating existing objects".

Replication will also stop if you remove the AWS Identity and Access Management (IAM) role, the AWS Key Management Service (AWS KMS) permissions, or the bucket policy permissions that grant Amazon S3 the required permissions. However, we don't recommend these approaches because they cause replication to fail. Amazon S3 reports the replication status for affected objects as `FAILED`. If permissions are later restored, objects marked as `FAILED` are *not* automatically replicated. To replicate those objects, you must use S3 Batch Replication.

Replicating existing objects with Batch Replication

S3 Batch Replication differs from live replication, which continuously and automatically replicates new objects across Amazon S3 buckets. Instead, S3 Batch Replication occurs on demand on existing objects. You can use S3 Batch Replication to replicate the following types of objects:

- Objects that existed before a replication configuration was in place

- Objects that have previously been replicated
- Objects that have failed replication

You can replicate these objects on demand by using a Batch Operations job.

To get started with Batch Replication, you can:

- **Initiate Batch Replication for a new replication rule or destination** – You can create a one-time Batch Replication job when you're creating the first rule in a new replication configuration or when you're adding a new destination bucket to an existing configuration through the Amazon S3 console.
- **Initiate Batch Replication for an existing replication configuration** – You can create a new Batch Replication job by using S3 Batch Operations through the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

When the Batch Replication job finishes, you receive a completion report. For more information about how to use this report to examine the job, see Tracking job status and completion reports.

S3 Batch Replication considerations

Before using S3 Batch Replication, review the following list of considerations:

- Your source bucket must have an existing replication configuration. To enable replication, see Setting up live replication overview and Examples for configuring live replication.
- If you have S3 Lifecycle configured for your bucket, we recommend disabling your lifecycle rules while the Batch Replication job is active. Doing so helps ensure parity between the source and destination buckets. Otherwise, these buckets could diverge, and the destination bucket won't be an exact replica of the source bucket. For example, consider the following scenario:
  - Your source bucket has multiple versions of an object and a delete marker on that object.
  - Your source and destination buckets have a lifecycle configuration to remove expired delete markers.

  In this scenario, Batch Replication might replicate the delete marker to the destination bucket before replicating the object versions. This behavior could result in your lifecycle configuration marking the delete marker as expired and the delete marker being removed from the destination bucket before the object versions are replicated.

- The AWS Identity and Access Management (IAM) role that you specify to run the Batch Operations job must have the necessary permissions to perform the underlying Batch Replication operation. For more information about creating IAM roles, see Configuring an IAM role for S3 Batch Replication.
- Batch Replication requires a manifest, which can be generated by Amazon S3. The generated manifest must be stored in the same AWS Region as the source bucket. If you choose not to generate the manifest, you can supply an Amazon S3 Inventory report or CSV file that contains the objects that you want to replicate. For more information, see the section called "Specifying a manifest for a Batch Replication job".
- If you recently added or updated the replication configuration on the source bucket, expect a delay of a few minutes before the change is fully propagated. We recommend waiting before creating a Batch Replication job with an S3-generated manifest.
- Batch Replication doesn't support re-replicating objects that were deleted by specifying the version ID of the object from the destination bucket. To re-replicate these objects, you can copy the source objects in place with a Batch Copy job. Copying those objects in place creates new versions of the objects in the source bucket and automatically initiates replication to the destination bucket. Deleting and recreating the destination bucket doesn't initiate replication.

  For more information about Batch Copy, see Examples that use Batch Operations to copy objects.
- If you're using a replication rule on the source bucket, make sure to update your replication configuration by granting the IAM role that's attached to the replication rule the proper permissions to replicate objects. This IAM role must have the necessary permissions to perform replication on both the source and destination buckets.
- If you submit multiple Batch Replication jobs for the same bucket within a short time frame, Amazon S3 runs those jobs concurrently.
- If you submit multiple Batch Replication jobs for two different buckets, be aware that Amazon S3 might not run all jobs concurrently. If you exceed the number of Batch Replication jobs that can run at one time on your account, Amazon S3 pauses the lower priority jobs to work on the higher priority ones. After the higher priority jobs are completed, any paused jobs become active again.
- Batch Replication isn't supported for objects that are stored in the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes.
- To batch replicate S3 Intelligent-Tiering objects that are stored in the Archive Access or Deep Archive Access storage tiers, you must first initiate a restore request and wait until the objects are moved to the Frequent Access tier.
- A single Batch Replication job can support a manifest with up to 20 billion objects.

- If you use S3 Batch Replication to replicate datasets cross region and your objects previously had their server-side encryption type updated from SSE-S3 to SSE-KMS, you may need additional permissions. On the source region bucket, you must have `kms:decrypt` permissions. Then, you will need the `kms:decrypt` and `kms:encrypt` permissions for the bucket in the destination region. For more information, see Replicating encrypted objects.
- If objects in the source bucket have annotations, the annotations are replicated independently alongside the object during Batch Replication. There are no separate job tasks for individual annotations. The job task status depends on both the parent object replication status and the annotation replication status.
- For best practices for managing incomplete multipart uploads in the destination bucket, see Managing incomplete multipart uploads in destination buckets.

Specifying a manifest for a Batch Replication job

A manifest is an Amazon S3 object that contains the object keys that you want Amazon S3 to act upon. If you want to create a Batch Replication job, you must supply either a user-generated manifest or have Amazon S3 generate a manifest based on your replication configuration.

If you supply a user-generated manifest, it must be in the form of an Amazon S3 Inventory report or a CSV file. If the objects in your manifest are in a versioned bucket, you must specify the version IDs for the objects. Only the object with the version ID that's specified in the manifest will be replicated. To learn more about specifying a manifest, see Specifying a manifest.

If you choose to have Amazon S3 generate a manifest file on your behalf, the objects listed use the same source bucket, prefix, and tags as your replication configurations on the source bucket. With a generated manifest, Amazon S3 replicates all eligible versions of your objects.

> **Note**
>
> If you choose to have Amazon S3 generate the manifest, the manifest must be stored in the same AWS Region as the source bucket.

Filters for a Batch Replication job

When creating your Batch Replication job, you can optionally specify additional filters, such as the object creation date and replication status, to reduce the scope of the job.

You can filter objects to replicate based on the `ObjectReplicationStatuses` value, by providing one or more of the following values:

- `"NONE"` – Indicates that Amazon S3 has never attempted to replicate the object before.
- `"FAILED"` – Indicates that Amazon S3 has attempted, but failed, to replicate the object before.
- `"COMPLETED"` – Indicates that Amazon S3 has successfully replicated the object before.
- `"REPLICA"` – Indicates that this object is a replica that Amazon S3 has replicated from another source bucket.

For more information about replication statuses, see Getting replication status information.

If you don't filter your Batch Replication job, Batch Operations attempts to replicate all objects (no matter their `ObjectReplicationStatus`) in your manifest that match the rules in your replication configuration, except for certain objects that aren't replicated by default. For more information, see the section called “What isn't replicated with replication configurations?”

Depending on your goal, you might set `ObjectReplicationStatuses` to one or more of the following values:

- To replicate only existing objects that have never been replicated, only include `"NONE"`.
- To retry replicating only objects that previously failed to replicate, only include `"FAILED"`.
- To both replicate existing objects and retry replicating objects that previously failed to replicate, include both `"NONE"` and `"FAILED"`.
- To backfill a destination bucket with objects that have been replicated to another destination, include `"COMPLETED"`.
- To replicate objects that were previously replicated, include `"REPLICA"`.

Batch Replication completion report

When you create a Batch Replication job, you can request a CSV completion report. This report shows the objects, replication success or failure codes, outputs, and descriptions. For more information about job tracking and completion reports, see Completion reports.

For a list of replication failure codes and descriptions, see Amazon S3 replication failure reasons.

For information about troubleshooting Batch Replication, see Batch Replication errors.

Getting started with Batch Replication

To learn more about how to use Batch Replication, see Create a Batch Replication job for new replication rules or destinations and Create a Batch Replication job for existing replication rules.

Configuring an IAM role for S3 Batch Replication

Because Amazon S3 Batch Replication is a type of Batch Operations job, you must create an AWS Identity and Access Management (IAM) role to grant Batch Operations permissions to perform actions on your behalf. You also must attach a Batch Replication IAM policy to the Batch Operations IAM role.

Use the following procedures to create a policy and an IAM role that give Batch Operations permission to initiate a Batch Replication job.

To create a policy for Batch Replication

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Under **Access management**, choose **Policies**.
3. Choose **Create policy**.
4. On the **Specify permissions** page, choose **JSON**.
5. Insert one of the following policies, depending on whether your manifest is generated by Amazon S3 or whether you are supplying your own manifest. For more information about manifests, see Specifying a manifest for a Batch Replication job.

   Before using these policies, replace the *`user input placeholders`* in the following policies with the names of your replication source bucket, manifest bucket, and completion report bucket.

   > **Note**
   >
   > Your IAM role for Batch Replication needs different permissions, depending on whether you are generating a manifest or supplying one, so make sure that you choose the appropriate policy from the following examples.

   #### Policy if using and storing an Amazon S3 generated manifest

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Action": [
            "s3:InitiateReplication"
         ],
         "Effect": "Allow",
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Action": [
            "s3:GetReplicationConfiguration",
            "s3:PutInventoryConfiguration"
         ],
         "Effect": "Allow",
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-source-bucket"
         ]
      },
      {
         "Action": [
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect": "Allow",
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
            "s3:PutObject"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*",
            "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
```

```
         ]
      }
   ]
}
```

Policy if using a user-supplied manifest

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Action": [
            "s3:InitiateReplication"
         ],
         "Effect": "Allow",
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-source-bucket/*"
         ]
      },
      {
         "Action": [
            "s3:GetObject",
            "s3:GetObjectVersion"
         ],
         "Effect": "Allow",
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-manifest-bucket/*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
            "s3:PutObject"
         ],
         "Resource": [
            "arn:aws:s3:::amzn-s3-demo-completion-report-bucket/*"
         ]
      }
   ]
```

```
}
```

6. Choose **Next**.
7. Specify a name for the policy, and then choose **Create policy**.

To create an IAM role for Batch Replication

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Under **Access management**, choose **Roles**.
3. Choose **Create role**.
4. Choose **AWS service** as the type of trusted entity. In the **Use case** section, choose **S3** as the service, and **S3 Batch Operations** as the use case.
5. Choose **Next**. The **Add permissions** page appears. In the search box, search for the policy that you created in the preceding procedure. Select the checkbox next to the policy name, then choose **Next**.
6. On the **Name, review, and create** page, specify a name for your IAM role.
7. In the **Step 1: Trust identities** section, verify that your IAM role is using the following trust policy:

   JSON

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"batchoperations.s3.amazonaws.com"
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

8. In the **Step 2: Add permissions** section, verify that your IAM role is using the policy that you created earlier.

9. Choose **Create role**.

Create a Batch Replication job for new replication rules or destinations

In Amazon S3, live replication doesn't replicate any objects that already existed in your source bucket before you created a replication configuration. Live replication automatically replicates only new and updated objects that are written to the bucket after the replication configuration is created. To replicate already existing objects, you can use S3 Batch Replication to replicate these objects on demand.

When you create the first rule in a new live replication configuration or add a new destination bucket to an existing replication configuration through the Amazon S3 console, you can optionally create a Batch Replication job. You can use this Batch Replication job to replicate existing objects in the source bucket to the destination bucket.

To use Batch Replication for an existing configuration without adding a new destination bucket, see Create a Batch Replication job for existing replication rules.

Prerequisites

Before creating your Batch Replication job, you must create a Batch Operations AWS Identity and Access Management (IAM) role to grant Amazon S3 permissions to perform actions on your behalf. For more information, see Configuring an IAM role for S3 Batch Replication.

Using Batch Replication for a new replication rule or destination through the Amazon S3 console

When you create the first rule in a new replication configuration or add a new destination bucket to an existing configuration through the Amazon S3 console, you can choose to create a Batch Replication job to replicate existing objects in the source bucket.

To create a Batch Replication job when creating or updating a replication configuration

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **General purpose buckets** list, choose the name of the bucket that contains the objects that you want to replicate.

4. To create a new replication rule or edit an existing rule, choose the **Management** tab, and scroll down to **Replication rules**:

   - To create a new replication rule, choose **Create replication rule**. For examples of how to set up a basic replication rule, see Examples for configuring live replication.
   - To edit an existing replication rule, select the option button next to the rule name, and then choose **Edit rule**.

5. Create your new replication rule or edit the destination for your existing replication rule, and choose **Save**.

   After you create the first rule in a new replication configuration or edit an existing configuration to add a new destination, a **Replicate existing objects?** dialog appears, giving you the option to create a Batch Replication job.

6. If you want to create and run this job now, choose **Yes, replicate existing objects**.

   If you want to create a Batch Replication job at a later time, choose **No, do not replicate existing objects**.

7. If you chose **Yes, replicate existing objects**, the **Create Batch Operations job** page appears. The S3 Batch Replication job has the following settings:

   Job run options

   If you want the S3 Batch Replication job to run immediately, choose **Automatically run the job when it's ready**. If you want to run the job at a later time, choose **Wait to run the job when it's ready**.

   > **Note**
   >
   > If you choose **Automatically run the job when it's ready**, you won't be able to create and save a Batch Operations manifest. To save the Batch Operations manifest, choose **Wait to run the job when it's ready**.

   Batch Operations manifest

   If you chose **Wait to run the job when it's ready**, the **Batch Operations manifest** section appears. The manifest is a list of all of the objects that you want to run the specified action on. You can choose to save the manifest. Similar to S3 Inventory files, the manifest will be

saved as a CSV file and stored in a bucket. To learn more about Batch Operations manifests, see Specifying a manifest.

Completion report

S3 Batch Operations executes one task for each object specified in the manifest. Completion reports provide an easy way to view the results of your tasks in a consolidated format with no additional setup required. You can request a completion report for all tasks or only for failed tasks. To learn more about completion reports, see Completion reports.

Permissions

One of the most common causes of replication failures is insufficient permissions in the provided AWS Identity and Access Management (IAM) role. For information about creating this role, see Configuring an IAM role for S3 Batch Replication. Make sure that you create or choose an IAM role that has the required permissions for Batch Replication.

8. Choose **Save**.

Create a Batch Replication job for existing replication rules

In Amazon S3, live replication doesn't replicate any objects that already existed in your source bucket before you created a replication configuration. Live replication automatically replicates only new and updated objects that are written to the bucket after the replication configuration is created. To replicate already existing objects, you can use S3 Batch Replication to replicate these objects on demand.

You can configure S3 Batch Replication for an existing replication configuration by using the AWS SDKs, AWS Command Line Interface (AWS CLI), or the Amazon S3 console. For an overview of Batch Replication, see Replicating existing objects with Batch Replication.

When the Batch Replication job finishes, you receive a completion report. For more information about how to use the report to examine the job, see Tracking job status and completion reports.

Prerequisites

Before creating your Batch Replication job, you must create a Batch Operations AWS Identity and Access Management (IAM) role to grant Amazon S3 permissions to perform actions on your behalf. For more information, see Configuring an IAM role for S3 Batch Replication.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. Choose **Create job**.
4. Verify that the **AWS Region** section shows the Region where you want to create your job.
5. In the **Manifest** section, specify the manifest format that you want to use. The manifest is a list of all of the objects that you want to run the specified action on. To learn more about Batch Operations manifests, see Specifying a manifest.

   - If you have a manifest prepared, choose **S3 inventory report (manifest.json)** or **CSV**. If your manifest is in a versioned bucket, you can specify the version ID for the manifest. If you don't specify a version ID, Batch Operations uses the current version of your manifest. For more information about creating a manifest, see Specifying a manifest.

     > **Note**
     >
     > If the objects in your manifest are in a versioned bucket, you must specify the version IDs for the objects. For more information, see Specifying a manifest.

   - To create a manifest based on your replication configuration, choose **Create manifest using S3 Replication configuration**. Then choose the source bucket of your replication configuration.
6. (Optional) If you chose **Create manifest using S3 Replication configuration**, you can include additional filters, such as the object creation date and replication status. For examples of how to filter by replication status, see Specifying a manifest for a Batch Replication job.
7. (Optional) If you chose **Create manifest using S3 Replication configuration**, you can save the generated manifest. To save this manifest, select **Save Batch Operations manifest**. Then specify the destination bucket for the manifest and choose whether to encrypt the manifest.

   > **Note**
   >
   > The generated manifest must be stored in the same AWS Region as the source bucket.

8. Choose **Next**.
9. On the **Operations** page, choose **Replicate**, then choose **Next**.

10. (Optional) Provide a **Description**.
11. Adjust the **Priority** of the job if needed. Higher numbers indicate higher priority. Amazon S3 attempts to run higher priority jobs before lower priority jobs. For more information about job priority, see Assigning job priority.
12. (Optional) Generate a completion report. To generate this report, select **Generate completion report**.

    If you choose to generate a completion report, you must choose either to report **Failed tasks only** or **All tasks**, and provide a destination bucket for the report.
13. In the **Permissions** section, make sure that you choose an IAM role that has the required permissions for Batch Replication. One of the most common causes of replication failures is insufficient permissions in the provided IAM role. For information about creating this role, see Configuring an IAM role for S3 Batch Replication.
14. (Optional) Add job tags to the Batch Replication job.
15. Choose **Next**.
16. Review your job configuration, and then choose **Create job**.

Using the AWS CLI with an S3 manifest

The following example `create-job` command creates an S3 Batch Replication job by using an S3 generated manifest for the AWS account *111122223333*. This example replicates existing objects and objects that previously failed to replicate. For information about filtering by replication status, see Specifying a manifest for a Batch Replication job.

To use this command, replace the *user input placeholders* with your own information. Replace the IAM role `role/`*batch-Replication-IAM-policy* with the IAM role that you previously created. For more information, see Configuring an IAM role for S3 Batch Replication.

```
aws s3control create-job --account-id 111122223333 \
--operation '{"S3ReplicateObject":{}}' \
--report '{"Bucket":"arn:aws:s3:::amzn-s3-demo-completion-report-bucket",\
"Prefix":"batch-replication-report", \
"Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
--manifest-generator '{"S3JobManifestGenerator": {"ExpectedBucketOwner":
 "111122223333", \
"SourceBucket": "arn:aws:s3:::amzn-s3-demo-source-bucket", \
"EnableManifestOutput": false, "Filter": {"EligibleForReplication": true, \
```

```
"ObjectReplicationStatuses": ["NONE","FAILED"]}}}' \
--priority 1 \
--role-arn arn:aws:iam::111122223333:role/batch-Replication-IAM-policy \
--no-confirmation-required \
--region source-bucket-region
```

> **Note**
>
> You must initiate the job from the same AWS Region as the replication source bucket.

After you have successfully initiated a Batch Replication job, you receive the job ID as the response. You can monitor this job by using the following `describe-job` command. To use this command, replace the *`user input placeholders`* with your own information.

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-bucket-region
```

Using the AWS CLI with a user-provided manifest

The following example creates an S3 Batch Replication job by using a user-defined manifest for AWS account *`111122223333`*. If the objects in your manifest are in a versioned bucket, you must specify the version IDs for the objects. Only the object with the version ID specified in the manifest will be replicated. For more information about creating a manifest, see Specifying a manifest.

To use this command, replace the *`user input placeholders`* with your own information. Replace the IAM role `role/`*`batch-Replication-IAM-policy`* with the IAM role that you previously created. For more information, see Configuring an IAM role for S3 Batch Replication.

```
aws s3control create-job --account-id 111122223333 \
--operation '{"S3ReplicateObject":{}}' \
--report '{"Bucket":"arn:aws:s3:::amzn-s3-demo-completion-report-bucket",\
"Prefix":"batch-replication-report", \
"Format":"Report_CSV_20180820","Enabled":true,"ReportScope":"AllTasks"}' \
--manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820",\
"Fields":["Bucket","Key","VersionId"]},\
"Location":{"ObjectArn":"arn:aws:s3:::amzn-s3-demo-manifest-bucket/manifest.csv",\
"ETag":"Manifest Etag"}}' \
--priority 1 \
--role-arn arn:aws:iam::111122223333:role/batch-Replication-IAM-policy \
```

```
--no-confirmation-required \
--region source-bucket-region
```

> **Note**
>
> You must initiate the job from the same AWS Region as the replication source bucket.

After you have successfully initiated a Batch Replication job, you receive the job ID as the response. You can monitor this job by using the following `describe-job` command.

```
aws s3control describe-job --account-id 111122223333 --job-id job-id --region source-bucket-region
```

Troubleshooting replication

This section lists troubleshooting tips for Amazon S3 Replication and information about S3 Batch Replication errors.

Topics

- Troubleshooting tips for S3 Replication
- Batch Replication errors

Troubleshooting tips for S3 Replication

If object replicas don't appear in the destination bucket after you configure replication, use these troubleshooting tips to identify and fix issues.

- The majority of objects replicate within 15 minutes. The time that it takes Amazon S3 to replicate an object depends on several factors, including the source and destination Region pair, and the size of the object. For large objects, replication can take up to several hours. For visibility into replication times, you can use S3 Replication Time Control (S3 RTC).

  If the object that is being replicated is large, wait a while before checking to see whether it appears in the destination. You can also check the replication status of the source object. If the object replication status is `PENDING`, Amazon S3 hasn't completed the replication. If the object replication status is `FAILED`, check the replication configuration that's set on the source bucket.

Additionally, to receive information about failures during replication, you can set up Amazon S3 Event Notifications replication to receive failure events. For more information, see Receiving replication failure events with Amazon S3 Event Notifications.

- To check the replication status of an object, you can call the `HeadObject` API operation. The `HeadObject` API operation returns the `PENDING`, `COMPLETED`, or `FAILED` replication status of an object. In a response to a `HeadObject` API call, the replication status is returned in the `x-amz-replication-status` header.

  > **Note**
  >
  > To run `HeadObject`, you must have read access to the object that you're requesting. A `HEAD` request has the same options as a `GET` request, without performing a `GET` operation. For example, to run a `HeadObject` request by using the AWS Command Line Interface (AWS CLI), you can run the following command. Replace the *`user input placeholders`* with your own information.
  >
  > ```
  > aws s3api head-object --bucket amzn-s3-demo-source-bucket --key index.html
  > ```

- If `HeadObject` returns objects with a `FAILED` replication status, you can use S3 Batch Replication to replicate those failed objects. For more information, see the section called "Replicating existing objects". Alternatively, you can re-upload the failed objects to the source bucket, which will initiate replication for the new objects.

  > **Note**
  >
  > If your replication rule uses the change object ownership to the destination bucket owner option and the object exists in the destination bucket but shows a `FAILED` replication status, S3 Batch Replication might not resolve this status. If you encounter this situation, contact AWS Support to verify the replication state of the affected objects.

- In the replication configuration on the source bucket, verify the following:
  - The Amazon Resource Name (ARN) of the destination bucket is correct.
  - The key name prefix is correct. For example, if you set the configuration to replicate objects with the prefix `Tax`, then only objects with key names such as `Tax/document1` or `Tax/document2` are replicated. An object with the key name `document3` is not replicated.

  - The status of the replication rule is `Enabled`.
- Verify that versioning hasn't been suspended on any bucket in the replication configuration. Both the source and destination buckets must have versioning enabled.
- If a replication rule is set to **Change object ownership to the destination bucket owner**, then the AWS Identity and Access Management (IAM) role that's used for replication must have the `s3:ObjectOwnerOverrideToBucketOwner` permission. This permission is granted on the resource (in this case, the destination bucket). For example, the following `Resource` statement shows how to grant this permission on the destination bucket:

```
{
  "Effect":"Allow",
  "Action":[
    "s3:ObjectOwnerOverrideToBucketOwner"
  ],
  "Resource":"arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
}
```

- If the destination bucket is owned by another account, the owner of the destination bucket must also grant the `s3:ObjectOwnerOverrideToBucketOwner` permission to the source bucket owner through the destination bucket policy. To use the following example bucket policy, replace the *`user input placeholders`* with your own information:

  JSON

```
{
  "Version":"2012-10-17",
  "Id": "Policy1644945280205",
  "Statement": [
    {
      "Sid": "Stmt1644945277847",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
      },
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateTags",
        "s3:ObjectOwnerOverrideToBucketOwner"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/*"
    }
```

```
  ]
}
```

> **Note**
>
> If the destination bucket's object ownership settings include **Bucket owner enforced**, then you don't need to update the setting to **Change object ownership to the destination bucket owner** in the replication rule. The object ownership change will occur by default. For more information about changing replica ownership, see Changing the replica owner.

- If you're setting the replication configuration in a cross-account scenario, where the source and destination buckets are owned by different AWS accounts, the destination buckets can't be configured as a Requester Pays bucket. For more information, see Using Requester Pays general purpose buckets for storage transfers and usage.
- If a bucket's source objects are encrypted by using server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), then the replication rule must be configured to include AWS KMS-encrypted objects. Make sure to select **Replicate objects encrypted with AWS KMS** under your **Encryption** settings in the Amazon S3 console. Then, select an AWS KMS key for encrypting the destination objects.

> **Note**
>
> If the destination bucket is in a different account, specify an AWS KMS customer managed key that is owned by the destination account. Don't use the default Amazon S3 managed key (`aws/s3`). Using the default key encrypts the objects with the Amazon S3 managed key that's owned by the source account, preventing the object from being shared with another account. As a result, the destination account won't be able to access the objects in the destination bucket.

To use an AWS KMS key that belongs to the destination account to encrypt the destination objects, the destination account must grant the `kms:GenerateDataKey` and `kms:Encrypt` permissions to the replication role in the KMS key policy. To use the following example statement in your KMS key policy, replace the *`user input placeholders`* with your own information:

```
{
    "Sid": "AllowS3ReplicationSourceRoleToUseTheKey",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::123456789101:role/s3-replication-role"
    },
    "Action": ["kms:GenerateDataKey", "kms:Encrypt"],
    "Resource": "*"
}
```

If you use an asterisk (*) for the `Resource` statement in the AWS KMS key policy, the policy grants permission to use the KMS key to only the replication role. The policy doesn't allow the replication role to elevate its permissions.

By default, the KMS key policy grants the root user full permissions to the key. These permissions can be delegated to other users in the same account. Unless there are `Deny` statements in the source KMS key policy, using an IAM policy to grant the replication role permissions to the source KMS key is sufficient.

> **Note**
>
> KMS key policies that restrict access to specific CIDR ranges, virtual private cloud (VPC) endpoints, or S3 access points can cause replication to fail.

If either the source or destination KMS keys grant permissions based on the encryption context, confirm that Amazon S3 Bucket Keys are turned on for the buckets. If the buckets have S3 Bucket Keys turned on, the encryption context must be the bucket-level resource, like this:

```
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::amzn-s3-demo-source-bucket"
     ]
"kms:EncryptionContext:arn:aws:arn": [
     "arn:aws:s3:::amzn-s3-demo-destination-bucket"
     ]
```

In addition to the permissions granted by the KMS key policy, the source account must add the following minimum permissions to the replication role's IAM policy:

```
{
    "Effect": "Allow",
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": [
        "Source-KMS-Key-ARN"
    ]
},
{
    "Effect": "Allow",
    "Action": [
        "kms:GenerateDataKey",
        "kms:Encrypt"
    ],
    "Resource": [
        "Destination-KMS-Key-ARN"
    ]
}
```

Important

If you use S3 Batch Replication to replicate datasets cross region and your objects previously had their server-side encryption type updated from SSE-S3 to SSE-KMS, you may need additional permissions. On the source region bucket, you must have `kms:decrypt` permissions. Then, you will need the `kms:decrypt` and `kms:encrypt` permissions for the bucket in the destination region.

For more information about how to replicate objects that are encrypted with AWS KMS, see Replicating encrypted objects.

- If the destination bucket is owned by another AWS account, verify that the bucket owner has a bucket policy on the destination bucket that allows the source bucket owner to replicate objects. For an example, see Configuring replication for buckets in different accounts.
- To use Object Lock with replication, you must grant two additional permissions on the source S3 bucket in the AWS Identity and Access Management (IAM) role that you use to set up replication. The two additional permissions are `s3:GetObjectRetention` and `s3:GetObjectLegalHold`.

If the role has an `s3:Get*` permission statement, that statement satisfies the requirement. For more information, see the section called "Using Object Lock with S3 Replication".

- If your objects still aren't replicating after you've validated the permissions, check for any explicit `Deny` statements in the following locations:
  - `Deny` statements in the source or destination bucket policies. Replication fails if the bucket policy denies access to the replication role for any of the following actions:

    Source bucket:

    ```
    "s3:GetReplicationConfiguration",
    "s3:ListBucket",
    "s3:GetObjectVersionForReplication",
    "s3:GetObjectVersionAcl",
    "s3:GetObjectVersionTagging"
    ```

    Destination buckets:

    ```
    "s3:ReplicateObject",
    "s3:ReplicateDelete",
    "s3:ReplicateTags"
    ```

  - `Deny` statements or permissions boundaries attached to the IAM role can cause replication to fail.
  - `Deny` statements in AWS Organizations service control policies (SCPs) that are attached to either the source or destination accounts can cause replication to fail.
  - `Deny` statements in AWS Organizations resource control policies (RCPs) that are attached to either the source or destination buckets can cause replication to fail.
- If an object replica doesn't appear in the destination bucket, the following issues might have prevented replication:
  - Amazon S3 doesn't replicate an object in a source bucket that is a replica created by another replication configuration. For example, if you set a replication configuration from bucket A to bucket B to bucket C, Amazon S3 doesn't replicate object replicas in bucket B to bucket C.
  - A source bucket owner can grant other AWS accounts permission to upload objects. By default, the source bucket owner doesn't have permissions for the objects created by other accounts. The replication configuration replicates only the objects for which the source bucket owner has access permissions. To avoid this problem, the source bucket owner can grant other AWS accounts permissions to create objects conditionally, requiring explicit access permissions on

those objects. For an example policy, see Grant cross-account permissions to upload objects while ensuring that the bucket owner has full control.

- Suppose that in the replication configuration, you add a rule to replicate a subset of objects that have a specific tag. In this case, you must assign the specific tag key and value at the time the object is created in order for Amazon S3 to replicate the object. If you first create an object and then add the tag to the existing object, Amazon S3 doesn't replicate the object.
- Use Amazon S3 Event Notifications to notify you of instances when objects don't replicate to their destination AWS Region. Amazon S3 Event Notifications are available through Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), or AWS Lambda. For more information, see Receiving replication failure events with Amazon S3 Event Notifications.

  You can also view replication failure reasons by using Amazon S3 Event Notifications. To review the list of failure reasons, see Amazon S3 replication failure reasons.

Batch Replication errors

To troubleshoot objects that aren't replicating to the destination bucket, check the different types of permissions for your buckets, replication role, and IAM role that's used to create the Batch Replication job. Also, make sure to check the Block Public Access settings and S3 Object Ownership settings for your buckets.

For additional troubleshooting tips for working with Batch Operations, see the section called "Troubleshooting S3 Batch Operations".

If you've set up replication and objects aren't replicating, see Why aren't my Amazon S3 objects replicating when I set up replication between my buckets? in the AWS re:Post Knowledge Center.

While using Batch Replication, you might encounter one of these errors:

- Manifest generation found no keys matching the filter criteria.

  This error occurs for one of the following reasons:

  - When objects in the source bucket are stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes.

    To use Batch Replication on these objects, first restore them to the S3 Standard storage class by using a **Restore** (`S3InitiateRestoreObjectOperation`) operation in a Batch

Operations job. For more information, see Restoring an archived object and Restore objects (Batch Operations). After you've restored the objects, you can replicate them by using a Batch Replication job.

- When the provided filter criteria doesn't match any valid objects in the source bucket.

  Verify and correct the filter criteria. For example, in the Batch Replication rule, the filter criteria is looking for all objects in the source bucket with the prefix `Tax/`. If the prefix name was entered inaccurately, with a slash in the beginning and the end `/Tax/` instead of only at the end, then no S3 objects were found. To resolve the error, correct the prefix, in this case, from `/Tax/` to `Tax/` in the replication rule.

- Batch operation status is failed with reason: The job report could not be written to your report bucket.

  This error occurs if the IAM role that's used for the Batch Operations job is unable to put the completion report into the location that was specified when you created the job. To resolve this error, check that the IAM role has the `s3:PutObject` permission for the bucket where you want to save the Batch Operations completion report. We recommend delivering the report to a bucket different from the source bucket.

- Batch operation is completed with failures and Total failed is not 0.

  This error occurs if there are insufficient object permissions issues with the Batch Replication job that is running. If you're using a replication rule for your Batch Replication job, make sure that the IAM role that's used for replication has the proper permissions to access objects from either the source or destination bucket. You can also check the Batch Replication completion report to review the specific Amazon S3 replication failure reason.

- Batch job ran successfully but the number of objects expected in destination bucket is not the same.

  This error occurs when there's a mismatch between the objects listed in the manifest that's supplied in the Batch Replication job and the filters that you selected when you created the job. You might also receive this message when the objects in your source bucket don't match any replication rules and aren't included in the generated manifest.

Batch Operations failures occur after adding a new replication rule to an existing replication configuration

Batch Operations attempts to perform existing object replication for every rule in the source bucket's replication configuration. If there are problems with any of the existing replication rules, failures might occur.

The Batch Operations job's completion report explains the job failure reasons. For a list of common errors, see Amazon S3 replication failure reasons.

Monitoring replication with metrics, event notifications, and statuses

You can monitor your live replication configurations and your S3 Batch Replication jobs through the following mechanisms:

- **S3 Replication metrics** – When you enable S3 Replication metrics, Amazon CloudWatch emits metrics that you can use to track bytes pending, operations pending, and replication latency at the replication rule level. You can view S3 Replication metrics through the Amazon S3 console and the Amazon CloudWatch console. In the Amazon S3 console, you can view these metrics in the source bucket's **Metrics** tab. For more information about S3 Replication metrics, see the section called “Using S3 Replication metrics”.
- **S3 Storage Lens metrics** – In addition to S3 Replication metrics, you can use the replication-related Data Protection metrics provided by S3 Storage Lens dashboards. For example, if you use the free metrics in S3 Storage Lens, you can see metrics such as the total number of bytes that are replicated from the source bucket or the count of replicated objects from the source bucket.

  To audit your overall replication stance, you can enable advanced metrics in S3 Storage Lens. With advanced metrics in S3 Storage Lens, you can see how many replication rules you have of various types, including the count of replication rules with a replication destination that's not valid.

  For more information about working with replication metrics in S3 Storage Lens, see the section called “Viewing replication metrics in S3 Storage Lens dashboards”.
- **S3 Event Notifications** – S3 Event Notifications can notify you at the object level in instances when objects don't replicate to their destination AWS Region or when objects aren't replicated within certain thresholds. S3 Event Notifications provides the following replication event types: `s3:Replication:OperationFailedReplication`, `s3:Replication:OperationMissedThreshold`,

`s3:Replication:OperationReplicatedAfterThreshold`, and `s3:Replication:OperationNotTracked`.

Amazon S3 events are available through Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), or AWS Lambda. For more information, see the section called "Receiving replication failure events".

- **Replication status values** – You can also retrieve the replication status of your objects. The replication status can help you determine the current state of an object that's being replicated. The replication status of a source object will return either `PENDING`, `COMPLETED`, or `FAILED`. The replication status of a replica will return `REPLICA`.

  You can also use replication status values when you're creating S3 Batch Replication jobs. For example, you can use these status values to replicate objects that have either never been replicated or that have failed replication.

  For more information about retrieving the replication status of your objects, see the section called "Getting replication status". For more information about using these values with Batch Replication, see the section called "Filters for a Batch Replication job".

Topics

- Using S3 Replication metrics
- Viewing replication metrics in S3 Storage Lens dashboards
- Receiving replication failure events with Amazon S3 Event Notifications
- Getting replication status information

Using S3 Replication metrics

S3 Replication metrics provide detailed metrics for the replication rules in your replication configuration. With replication metrics, you can monitor minute-by-minute progress by tracking bytes pending, operations pending, operations that failed replication, and replication latency.

Note

- S3 Replication metrics are billed at the same rate as Amazon CloudWatch custom metrics. For more information, see Amazon CloudWatch pricing.

- If you're using S3 Replication Time Control, Amazon CloudWatch begins reporting replication metrics 15 minutes after you enable S3 RTC on the respective replication rule.

S3 Replication metrics are turned on automatically when you enable S3 Replication Time Control (S3 RTC). You can also enable S3 Replication metrics independently of S3 RTC while creating or editing a rule. S3 RTC includes other features, such as a service level agreement (SLA) and notifications for missed thresholds. For more information, see Meeting compliance requirements with S3 Replication Time Control.

When S3 Replication metrics are enabled, Amazon S3 publishes the following metrics to Amazon CloudWatch. CloudWatch metrics are delivered on a best-effort basis.

Metric name	Metric description	Which objects does this metric apply to?	Which Region is this metric published in?	Is this metric still published if the destination bucket is deleted?	Is this metric still published if replication doesn't occur?
Bytes Pending Replication	The total number of bytes of objects that are pending replication for a given replication rule.	This metric applies only to new objects that are replicated with S3 Cross-Region Replication (S3 CRR) or S3 Same-Region Replication (S3 SRR).	This metric is published in the Region of the destination bucket.	No	Yes
Replication Latency	The maximum number of	This metric applies only to new	This metric is published in the Region of	No	Yes

Metric name	Metric description	Which objects does this metric apply to?	Which Region is this metric published in?	Is this metric still published if the destination bucket is deleted?	Is this metric still published if replication doesn't occur?
	seconds by which the replication destination bucket is behind the source bucket for a given replication rule.	objects that are replicated with S3 CRR or S3 SRR.	the destination bucket.		

Metric name	Metric description	Which objects does this metric apply to?	Which Region is this metric published in?	Is this metric still published if the destination bucket is deleted?	Is this metric still published if replication doesn't occur?
Operations Pending Replication	The number of operations that are pending replication for a given replication rule. This metric tracks operations related to objects, delete markers, tags, access control lists (ACLs), and S3 Object Lock.	This metric applies only to new objects that are replicated with S3 CRR or S3 SRR.	This metric is published in the Region of the destination bucket.	No	Yes

Metric name	Metric description	Which objects does this metric apply to?	Which Region is this metric published in?	Is this metric still published if the destination bucket is deleted?	Is this metric still published if replication doesn't occur?
Operations Failed Replication	The number of operations that failed replication for a given replication rule. This metric tracks operations related to objects, delete markers, tags, access control lists (ACLs), and Object Lock. **Operations Failed Replication** tracks S3 Replication failures aggregated at a per-minute interval. To identify the specific	This metric applies both to new objects that are replicated with S3 CRR or S3 SRR and also to existing objects that are replicated with S3 Batch Replication. **Note** If an S3 Batch Replication job fails to run at all, metrics aren't sent	This metric is published in the Region of the source bucket.	Yes	No

Metric name	Metric description	Which objects does this metric apply to?	Which Region is this metric published in?	Is this metric still published if the destination bucket is deleted?	Is this metric still published if replication doesn't occur?
	objects that have failed replication and their failure reasons, subscribe to the `Operation FailedReplication` event in Amazon S3 Event Notifications. For more information, see Receiving replication failure events with Amazon S3 Event Notifications.	to Amazon CloudWatch. For example, your job won't run if you don't have the necessary permissions to run an S3 Batch Replication job, or if the tags or prefix in your replicati			

Metric name	Metric description	Which objects does this metric apply to?	Which Region is this metric published in?	Is this metric still published if the destination bucket is deleted?	Is this metric still published if replication doesn't occur?
		on configuration don't match.			

For information about working with these metrics in CloudWatch, see the section called "S3 Replication metrics in CloudWatch".

Enabling S3 Replication metrics

You can start using S3 Replication metrics with a new or existing replication rule. For full instructions on creating replication rules, see Configuring replication for buckets in the same account. You can choose to apply your replication rule to an entire S3 bucket, or to Amazon S3 objects with a specific prefix or tag.

This topic provides instructions for enabling S3 Replication metrics in your replication configuration when the source and destination buckets are owned by the same or different AWS accounts.

To enable replication metrics by using the AWS Command Line Interface (AWS CLI), you must add a replication configuration to the source bucket with `Metrics` enabled. In this example configuration, objects under the prefix *`Tax`* are replicated to the destination bucket *`amzn-s3-demo-bucket`*, and metrics are generated for those objects.

```
{
    "Rules": [
        {
            "Status": "Enabled",
            "Filter": {
                "Prefix": "Tax"
```

```
            },
            "Destination": {
                "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
                "Metrics": {
                    "Status": "Enabled"
                }
            },
            "Priority": 1
        }
    ],
    "Role": "IAM-Role-ARN"
}
```

Viewing replication metrics

You can view S3 Replication metrics in the source general purpose bucket's **Metrics** tab in the Amazon S3 console. These Amazon CloudWatch metrics are also available in the Amazon CloudWatch console. When you enable S3 Replication metrics, Amazon CloudWatch emits metrics that you can use to track bytes pending, operations pending, and replication latency at the replication rule level.

S3 Replication metrics are turned on automatically when you enable replication with S3 Replication Time Control (S3 RTC) by using the Amazon S3 console or the Amazon S3 REST API. You can also enable S3 Replication metrics independently of S3 RTC while creating or editing a rule.

If you're using S3 Replication Time Control, Amazon CloudWatch begins reporting replication metrics 15 minutes after you enable S3 RTC on the respective replication rule. For more information, see Using S3 Replication metrics.

Replication metrics track the rule IDs of the replication configuration. A replication rule ID can be specific to a prefix, a tag, or a combination of both.

For more information about CloudWatch metrics for Amazon S3, see Monitoring metrics with Amazon CloudWatch.

Prerequisites

Create a replication rule that has S3 Replication metrics enabled. For more information, see the section called “Enabling replication metrics”.

To view S3 Replication metrics through the source bucket's Metrics tab

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the source bucket that contains the objects that you want replication metrics for.
4. Choose the **Metrics** tab.
5. Under **Replication metrics**, choose the replication rules that you want to see metrics for.
6. Choose **Display charts**.

   Amazon S3 displays **Replication latency**, **Bytes pending replication**, **Operations pending replication**, and **Operations failed replication** charts for the rules that you selected.

Viewing replication metrics in S3 Storage Lens dashboards

In addition to S3 Replication metrics, you can use the replication-related Data Protection metrics provided by S3 Storage Lens. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Using S3 Storage Lens to protect your data.

S3 Storage Lens has two tiers of metrics: free metrics, and advanced metrics and recommendations, which you can upgrade to for an additional charge. With advanced metrics and recommendations, you can access additional metrics and features for gaining insight into your storage. For information about S3 Storage Lens pricing, see Amazon S3 pricing.

If you use the free metrics in S3 Storage Lens, you can see metrics such as the total number of bytes that are replicated from the source bucket or the count of replicated objects from the source bucket.

To audit your overall replication stance, you can enable advanced metrics in S3 Storage Lens. With advanced metrics in S3 Storage Lens, you can see how many replication rules you have of various types, including the count of replication rules with a replication destination that's not valid.

For a complete list of S3 Storage Lens metrics, including which replication metrics are in each tier, see the S3 Storage Lens metrics glossary.

Prerequisites

Create a live replication configuration or an S3 Batch Replication job.

To view replication metrics in Amazon S3 Storage Lens

1. Create an S3 Storage Lens dashboard. For step-by-step instructions, see the section called "Using the S3 console".
2. (Optional) During your dashboard setup, if you want to see all S3 Storage Lens replication metrics, select **Advanced metrics and recommendations** and then select **Advanced data protection metrics**. For a complete list of metrics, see the S3 Storage Lens metrics glossary.

   If you enable advanced metrics and recommendations, you can gain further insights into your replication configurations. For example, you can use S3 Storage Lens replication rule count metrics to get detailed information about your buckets that are configured for replication. This information includes replication rules within and across buckets and Regions. For more information, see the section called "Count the total number of replication rules for each bucket".
3. After you've created your dashboard, open the dashboard, and choose the **Buckets** tab.
4. Scroll down to the **Buckets** section. Under **Metrics categories**, choose **Data protection**. Then clear **Summary**.
5. To filter the **Buckets** list to display only replication metrics, choose the **Preferences** icon ().
6. Clear the toggles for all data-protection metrics until only the replication metrics remain selected.
7. (Optional) Under **Page size**, choose the number of buckets to display in the list.
8. Choose **Continue**.

Receiving replication failure events with Amazon S3 Event Notifications

If you've enabled S3 Replication metrics on your replication configuration, you can set up Amazon S3 Event Notifications to notify you when objects don't replicate to their destination AWS Region. If you've enabled S3 Replication Time Control (S3 RTC) on your replication configuration, you can also be notified when objects don't replicate within the 15-minute S3 RTC threshold for replication.

Important

To receive replication failure event notifications (`s3:Replication:OperationFailedReplication`), you must enable S3 Replication metrics on your replication rule. You do not need to enable S3 Replication Time Control (S3 RTC) to receive failure notifications. S3 RTC is only required for threshold-based notifications (`s3:Replication:OperationMissedThreshold` and `s3:Replication:OperationReplicatedAfterThreshold`). You can enable S3 Replication metrics independently of S3 RTC. For more information, see Enabling S3 Replication metrics.

By using the following `Replication` event types, you can monitor the minute-by-minute progress of replication events by tracking bytes pending, operations pending, and replication latency. For more information about S3 Replication metrics, see Using S3 Replication metrics.

- The `s3:Replication:OperationFailedReplication` event type notifies you when an object that was eligible for replication failed to replicate.
- The `s3:Replication:OperationMissedThreshold` event type notifies you when an object that was eligible for replication that uses S3 RTC exceeds the 15-minute threshold for replication.
- The `s3:Replication:OperationReplicatedAfterThreshold` event type notifies you when an object that was eligible for replication that uses S3 RTC replicates after the 15-minute threshold.
- The `s3:Replication:OperationNotTracked` event type notifies you when an object that was eligible for live replication (either Same-Region Replication [SRR] or Cross-Region Replication [CRR]) is no longer being tracked by replication metrics.

For full descriptions of all the supported replication event types, see the section called "Supported event types for SQS, SNS, and Lambda".

For a list of the failure codes captured by S3 Event Notifications, see Amazon S3 replication failure reasons.

You can receive S3 Event Notifications through Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), or AWS Lambda. For more information, see the section called "Amazon S3 Event Notifications".

For instructions on how to configure Amazon S3 Event Notifications, see Enabling event notifications.

The following is an example of a message that Amazon S3 sends to publish an `s3:Replication:OperationFailedReplication` event. For more information, see the section called "Event message structure".


```
{
  "Records": [
    {
      "eventVersion": "2.2",
      "eventSource": "aws:s3",
      "awsRegion": "us-east-1",
      "eventTime": "2024-09-05T21:04:32.527Z",
      "eventName": "Replication:OperationFailedReplication",
      "userIdentity": {
        "principalId": "s3.amazonaws.com"
      },
      "requestParameters": {
        "sourceIPAddress": "s3.amazonaws.com"
      },
      "responseElements": {
        "x-amz-request-id": "123bf045-2b4b-4ca8-a211-c34a63c59426",
        "x-amz-id-2":
 "12VAWNDIHnwJsRhTccqQTeAPoXQmRt22KkewMV8G3XZihAuf9CLDdmkApgZzudaIe2KlLfDqGS0="
      },
      "s3": {
        "s3SchemaVersion": "1.0",
        "configurationId": "ReplicationEventName",
        "bucket": {
          "name": "amzn-s3-demo-bucket1",
          "ownerIdentity": {
            "principalId": "111122223333"
          },
          "arn": "arn:aws:s3:::amzn-s3-demo-bucket1"
        },
        "object": {
          "key": "replication-object-put-test.png",
          "size": 520080,
          "eTag": "e12345ca7e88a38428305d3ff7fcb99f",
          "versionId": "abcdeH0Xp66ep__QDjR76LK7Gc9X4wKO",
          "sequencer": "0066DA1CBF104C0D51"
        }
```

```
      },
      "replicationEventData": {
        "replicationRuleId": "notification-test-replication-rule",
        "destinationBucket": "arn:aws:s3:::amzn-s3-demo-bucket2",
        "s3Operation": "OBJECT_PUT",
        "requestTime": "2024-09-05T21:03:59.168Z",
        "failureReason": "AssumeRoleNotPermitted"
      }
    }
  ]
}
```

Amazon S3 replication failure reasons

The following table lists Amazon S3 Replication failure reasons. You can view these reasons by receiving the `s3:Replication:OperationFailedReplication` event with Amazon S3 Event Notifications and then looking at the `failureReason` value.

You can also view these failure reasons in an S3 Batch Replication completion report. For more information, see Batch Replication completion report.

Replication failure reason	Description
`AssumeRoleNotPermitted`	Amazon S3 can't assume the AWS Identity and Access Management (IAM) role that's specified in the replication configuration or in the Batch Operations job.
`DstBucketInvalidRegion`	The destination bucket is not in the same AWS Region as specified by the Batch Operations job. This error is specific to Batch Replication.
`DstBucketNotFound`	Amazon S3 is unable to find the destinati on bucket that's specified in the replication configuration.
`DstBucketObjectLockConfigMissing`	To replicate objects from a source bucket with Object Lock enabled, the destination bucket must also have Object Lock enabled.

Replication failure reason	Description
	This error indicates that Object Lock might not be enabled in the destination bucket. For more information, see Object Lock considerations.
`DstBucketUnversioned`	Versioning is not enabled for the S3 destination bucket. To replicate objects with S3 Replication, enable versioning for the destination bucket.
`DstDelObjNotPermitted`	Amazon S3 is unable to replicate delete markers to the destination bucket. The `s3:ReplicateDelete` permission might be missing for the destination bucket.
`DstKmsKeyInvalidState`	The AWS Key Management Service (AWS KMS) key for the destination bucket isn't in a valid state. Review and enable the required AWS KMS key. For more information about managing AWS KMS keys, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.
`DstKmsKeyNotFound`	The AWS KMS key that's configured for the destination bucket in the replication configuration doesn't exist.
`DstMultipartCompleteNotPermitted`	Amazon S3 is unable to complete multipart uploads of objects in the destination bucket. The `s3:ReplicateObject` permission might be missing for the destination bucket.
`DstMultipartInitNotPermitted`	Amazon S3 is unable to initiate multipart uploads of objects to the destination bucket. The `s3:ReplicateObject` permission might be missing for the destination bucket.

Replication failure reason	Description
DstMultipartUploadNotPermitted	Amazon S3 is unable to upload multipart upload objects to the destination bucket. The s3:ReplicateObject permission might be missing for the destination bucket.
DstObjectHardDeleted	S3 Batch Replication does not support re-replicating objects deleted with the version ID of the object from the destination bucket. This error is specific to Batch Replication.
DstPutAclNotPermitted	Amazon S3 is unable to replicate object access control lists (ACLs) to the destination bucket. The s3:ReplicateObject permission might be missing for the destination bucket.
DstPutLegalHoldNotPermitted	Amazon S3 is unable to put an Object Lock legal hold on the destination objects when it's replicating immutable objects. The s3:PutObjectLegalHold permission might be missing for the destination bucket. For more information, see Legal holds.
DstPutObjectNotPermitted	Amazon S3 is unable to replicate objects to the destination bucket. This can occur when required permissions (s3:ReplicateObject or s3:ObjectOwnerOverrideToBucketOwner permissions) are missing for the destination bucket or when the AWS KMS key policy doesn't allow the source bucket's replication role to use the AWS KMS key (kms:Decrypt and kms:GenerateDataKey* actions) at the destination bucket.

Replication failure reason	Description
DstPutRetentionNotPermitted	Amazon S3 is unable to put a retention period on the destination objects when it's replicating immutable objects. The s3:PutObjectRetention permission might be missing for the destination bucket.
DstPutTaggingNotPermitted	Amazon S3 is unable to replicate object tags to the destination bucket. The s3:ReplicateObject permission might be missing for the destination bucket.
DstVersionNotFound	Amazon S3 is unable to find the required object version in the destination bucket for which metadata needs to be replicated.
InitiateReplicationNotPermitted	Amazon S3 is unable to initiate replication on objects. The s3:InitiateReplication permission might be missing for the Batch Operations job. This error is specific to Batch Replication.
SrcBucketInvalidRegion	The source bucket isn't in the same AWS Region as specified by the Batch Operations job. This error is specific to Batch Replication.
SrcBucketNotFound	Amazon S3 is unable to find the source bucket.
SrcBucketReplicationConfigMissing	Amazon S3 couldn't find a replication configuration for the source bucket.

Replication failure reason	Description
SrcGetAclNotPermitted	Amazon S3 is unable to access the object in the source bucket for replication. The s3:GetObjectVersionAcl permission might be missing for the source bucket object. The objects in the source bucket must be owned by the bucket owner. If ACLs are enabled, then verify if Object Ownership is set to Bucket owner preferred or Object writer. If Object Ownership is set to Bucket owner preferred, then the source bucket objects must have the bucket-owner-full-control ACL for the bucket owner to become the object owner. The source account can take ownership of all objects in their bucket by setting Object Ownership to Bucket owner enforced and disabling ACLs.
SrcGetLegalHoldNotPermitted	Amazon S3 is unable to access the S3 Object Lock legal hold information.
SrcGetObjectNotPermitted	Amazon S3 is unable to access the object in the source bucket for replication. The s3:GetObjectVersionForReplication permission might be missing for the source bucket.
SrcGetRetentionNotPermitted	Amazon S3 is unable to access the S3 Object Lock retention period information.
SrcGetTaggingNotPermitted	Amazon S3 is unable to access object tag information from the source bucket. The s3:GetObjectVersionTagging permission might be missing for the source bucket.

Replication failure reason	Description
SrcHeadObjectNotPermitted	Amazon S3 is unable to retrieve object metadata from the source bucket. The s3:GetObjectVersionForReplication permission might be missing for the source bucket.
SrcKeyNotFound	Amazon S3 is unable to find the source object key to replicate. Source object may have been deleted before replication was complete.
SrcKmsKeyInvalidState	The AWS KMS key for the source bucket isn't in a valid state. Review and enable the required AWS KMS key. For more information about managing AWS KMS keys, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*.
SrcObjectNotEligible	Some objects aren't eligible for replication. This may be due to the object's storage class or the object tags don't match the replication configuration.
SrcObjectNotFound	Source object does not exist.
SrcReplicationNotPending	Amazon S3 has already replicated this object. This object is no longer pending replication.
SrcVersionNotFound	Amazon S3 is unable to find the source object version to replicate. Source object version may have been deleted before replication was complete.

Replication failure reason	Description
SrcAnnotationNotFound	Source annotation was not found. The annotation was deleted before replication completed.
SrcGetObjectAnnotationNotPermitted	Access denied when retrieving the source annotation for replication. The IAM role is missing s3:GetObjectVersionAnnotationForReplication .
DstPutObjectAnnotationNotPermitted	Access denied when replicating the annotation to the destination bucket. The IAM role is missing s3:ReplicateObject or does not have access to the required AWS KMS key.
SrcAnnotationEncryptionUnsupported	The object annotation encryption type is not supported.
AnnotationLimitExceeded	The object annotation count exceeds the limit (1000). Replication would exceed the annotation limit on the destination.

Related topics

Setting up permissions for live replication

Troubleshooting replication

Getting replication status information

Replication status can help you determine the current state of an object being replicated. The replication status of a source object will return either PENDING, COMPLETED, or FAILED. The replication status of a replica will return REPLICA.

You can also use replication status values when you're creating S3 Batch Replication jobs. For example, you can use these status values to replicate objects that have either never been replicated or that have failed replication. For more information about using these values with Batch

Replication, see the section called "Using replication status information with Batch Replication jobs".

Topics

- Replication status overview
- Replication status if replicating to multiple destination buckets
- Replication status if Amazon S3 replica modification sync is enabled
- Using replication status information with Batch Replication jobs
- Finding replication status

Replication status overview

In replication, you have a source bucket on which you configure replication and one or more destination buckets where Amazon S3 replicates objects. When you request an object (by using `GetObject`) or object metadata (by using `HeadObject`) from these buckets, Amazon S3 returns the `x-amz-replication-status` header in the response:

- When you request an object from the source bucket, Amazon S3 returns the `x-amz-replication-status` header if the object in your request is eligible for replication.

  For example, suppose that you specify the object prefix `TaxDocs` in your replication configuration to tell Amazon S3 to replicate only objects with the key name prefix `TaxDocs`. Any objects that you upload that have this key name prefix—for example, `TaxDocs/document1.pdf`—will be replicated. For object requests with this key name prefix, Amazon S3 returns the `x-amz-replication-status` header with one of the following values for the object's replication status: `PENDING`, `COMPLETED`, or `FAILED`.

  > **Note**
  >
  > If object replication fails after you upload an object, you can't retry replication. You must upload the object again, or you must use S3 Batch Replication to replicate any failed objects. S3 Lifecycle blocks expiration and transition actions on objects with `PENDING` or `FAILED` replication status. For more information about using Batch Replication, see the section called "Replicating existing objects".
  >
  > Objects transition to a `FAILED` state for issues such as missing replication role permissions, AWS Key Management Service (AWS KMS) permissions, or bucket permissions. For temporary failures, such as if a bucket or Region is unavailable,

replication status doesn't transition to `FAILED`, but remains `PENDING`. After the resource is back online, Amazon S3 resumes replicating those objects.

- When you request an object from a destination bucket, if the object in your request is a replica that Amazon S3 created, Amazon S3 returns the `x-amz-replication-status` header with the value `REPLICA`.

> **Note**
>
> Before deleting an object from a source bucket that has replication enabled, check the object's replication status to make sure that the object has been replicated.
> If an S3 Lifecycle configuration is enabled on the source bucket, Amazon S3 suspends lifecycle actions until it marks the object's replication status as `COMPLETED`. If replication status is `FAILED`, S3 Lifecycle continues to block expiration and transition actions on the object until you resolve the underlying replication issue. For more information, see S3 Lifecycle and S3 Replication.

Replication status if replicating to multiple destination buckets

When you replicate objects to multiple destination buckets, the `x-amz-replication-status` header acts differently. The header of the source object returns a value of `COMPLETED` only when replication is successful to all destinations. The header remains at the `PENDING` value until replication has completed for all destinations. If one or more destinations fail replication, the header returns `FAILED`.

Replication status if Amazon S3 replica modification sync is enabled

When your replication rules enable Amazon S3 replica modification sync, replicas can report statuses other than `REPLICA`. If metadata changes are in the process of replicating, the `x-amz-replication-status` header returns `PENDING`. If replica modification sync fails to replicate metadata, the header returns `FAILED`. If metadata is replicated correctly, the replicas return the header `REPLICA`.

Using replication status information with Batch Replication jobs

When creating a Batch Replication job, you can optionally specify additional filters, such as the object creation date and replication status, to reduce the scope of the job.

You can filter objects to replicate based on the `ObjectReplicationStatuses` value, by providing one or more of the following values:

- `"NONE"` – Indicates that Amazon S3 has never attempted to replicate the object before.
- `"FAILED"` – Indicates that Amazon S3 has attempted, but failed, to replicate the object before.
- `"COMPLETED"` – Indicates that Amazon S3 has successfully replicated the object before.
- `"REPLICA"` – Indicates that this is a replica object that Amazon S3 has replicated from another source.

For more information about using these replication status values with Batch Replication, see the section called “Filters for a Batch Replication job”.

Finding replication status

To get the replication status of the objects in a bucket, you can use the Amazon S3 Inventory tool. Amazon S3 sends a CSV file to the destination bucket that you specify in the inventory configuration. You can also use Amazon Athena to query the replication status in the inventory report. For more information about Amazon S3 Inventory, see Cataloging and analyzing your data with S3 Inventory.

You can also find the object replication status by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), or the AWS SDK.

Using the S3 console

In the Amazon S3 console, you can view the replication status for an object on the object's details page.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **General purpose buckets** list, choose the name of the replication source bucket.
4. In the **Objects** list, choose the object name. The object's details page appears.
5. On the **Properties** tab, scroll down to the **Object management overview** section. Under **Management configurations**, see the value under **Replication status**.

Using the AWS CLI

Use the AWS Command Line Interface (AWS CLI) `head-object` command to retrieve object metadata, as shown in the following example. Replace the *amzn-s3-demo-source-bucket1* with the name of your replication source bucket, and replace the other *user input placeholders* with your own information.

```
aws s3api head-object --bucket amzn-s3-demo-source-bucket1 --key object-key --version-id object-version-id
```

The command returns object metadata, including the `ReplicationStatus` as shown in the following example response.

```
{
   "AcceptRanges":"bytes",
   "ContentType":"image/jpeg",
   "LastModified":"Mon, 23 Mar 2015 21:02:29 GMT",
   "ContentLength":3191,
   "ReplicationStatus":"COMPLETED",
   "VersionId":"jfnW.HIMOfYiD_9rGbSkmroXsFj3fqZ.",
   "ETag":"\"6805f2cfc46c0f04559748bb039d69ae\"",
   "Metadata":{

   }
}
```

Using the AWS SDKs

The following code fragments get your replication status by using the AWS SDK for Java and AWS SDK for .NET, respectively.

Java

```
GetObjectMetadataRequest metadataRequest = new GetObjectMetadataRequest(bucketName,
 key);
ObjectMetadata metadata = s3Client.getObjectMetadata(metadataRequest);

System.out.println("Replication Status : " +
 metadata.getRawMetadataValue(Headers.OBJECT_REPLICATION_STATUS));
```

.NET

```
GetObjectMetadataRequest getmetadataRequest = new GetObjectMetadataRequest
    {
         BucketName = sourceBucket,
         Key        = objectKey
    };

GetObjectMetadataResponse getmetadataResponse =
 client.GetObjectMetadata(getmetadataRequest);
Console.WriteLine("Object replication status: {0}",
 getmetadataResponse.ReplicationStatus);
```

You can find the replication status of annotations by using the Amazon S3 API or Amazon S3 Metadata. The `ListObjectAnnotations` and `GetObjectAnnotation` API operations return the replication status for each annotation in the response. If you have an Amazon S3 Metadata table configuration that includes annotations, you can query the `replication_status` column in the annotation table. For more information, see the section called “Annotation table schema”.

Managing multi-Region traffic with Multi-Region Access Points

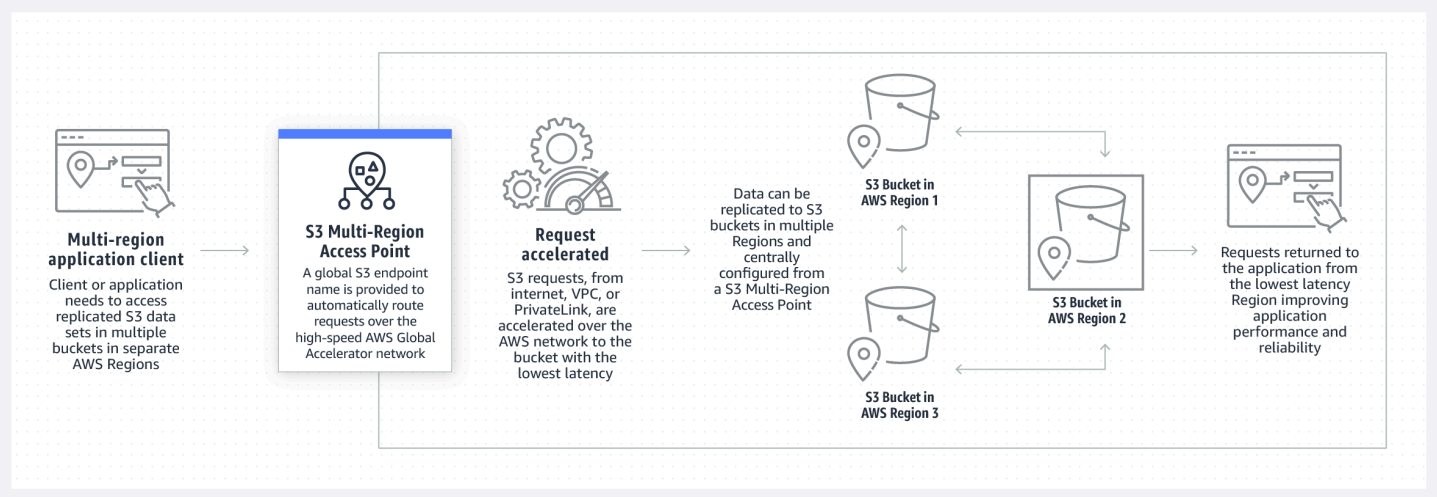
Amazon S3 Multi-Region Access Points provides a global endpoint that applications can use to fulfill requests from S3 buckets that are located in multiple AWS Regions. You can use Multi-Region Access Points to build multi-Region applications with the same architecture that's used in a single Region, and then run those applications anywhere in the world. Instead of sending requests over the congested public internet, Multi-Region Access Points provides built-in network resilience with acceleration of internet-based requests to Amazon S3. Application requests made to a Multi-Region Access Point global endpoint uses AWS Global Accelerator to automatically route over the AWS global network to the closest proximity S3 bucket with an active routing status.

If a Regional traffic disruption occurs, you can use Multi-Region Access Points failover controls to shift the S3 data request traffic between AWS Regions and redirect S3 traffic away from the disruptions within minutes. You can also test the application resiliency against a disruption to conduct application failover and perform disaster recovery simulations. If you need to connect and accelerate requests to S3 from outside of a VPC, you can simplify applications and network architecture with Amazon S3 Multi-Region Access Points. Your Multi-Region Access Points requests will be routed over the AWS global network and then back to S3 within the AWS Region, without having to traverse the public internet. As a result, you can build more highly available applications.

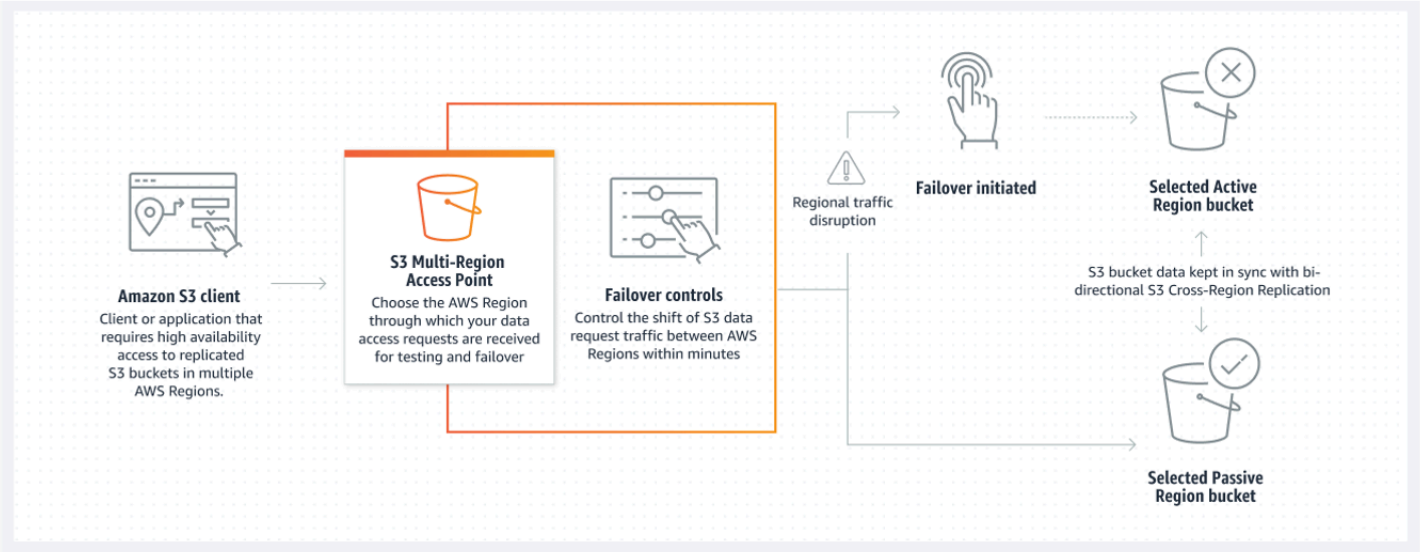
During your Multi-Region Access Points creation and setup, you'll specify a set of AWS Regions where you want to store data to be served through that Multi-Region Access Point. You can use the provided Multi-Region Access Points endpoint name to connect your clients. After you've established your client connections, you can select the existing or new buckets that you'd like to route the Multi-Region Access Points requests between. Then, use S3 Cross-Region Replication (CRR) rules to synchronize data among buckets in those Regions.

After you've set up your Multi-Region Access Point. you can then request or write data through the Multi-Region Access Points global endpoint. Amazon S3 automatically serves requests to the replicated data set from the closest available Region. Within the AWS Management Console, you're also able to view the underlying replication topology and replication metrics related to your Multi-Region Access Points requests. This gives you an even easier way to build, manage, and monitor storage for multi-Region applications. Alternatively, you can use Amazon CloudFront to automate the creation and configuration of S3 Multi-Region Access Points.

The following image is a graphical representation of an Amazon S3 Multi-Region Access Point in an active-active configuration. The graphic shows how Amazon S3 requests are automatically routed to buckets in the closest active AWS Region.



The following image is a graphical representation of an Amazon S3 Multi-Region Access Point in an active-passive configuration. The graphic shows how you can control Amazon S3 data-access traffic to fail over between active and passive AWS Regions.



Topics

- Creating Multi-Region Access Points
- Configuring a Multi-Region Access Point for use with AWS PrivateLink
- Making requests through a Multi-Region Access Point

Creating Multi-Region Access Points

To create a Multi-Region Access Point in Amazon S3, you do the following:

- Specify the name for the Multi-Region Access Point.
- Choose one bucket in each AWS Region that you want to serve requests for the Multi-Region Access Point.
- Configure the Amazon S3 Block Public Access settings for the Multi-Region Access Point.

You provide all of this information in a create request, which Amazon S3 processes asynchronously. Amazon S3 provides a token that you can use to monitor the status of the asynchronous creation request.

Make sure to resolve security warnings, errors, general warnings, and suggestions from AWS Identity and Access Management Access Analyzer before you save your policy. IAM Access Analyzer runs policy checks to validate your policy against IAM policy grammar and best practices. These checks generate findings and provide actionable recommendations to help you author policies that

are functional and conform to security best practices. To learn more about validating policies using IAM Access Analyzer, see IAM Access Analyzer policy validation in the *IAM User Guide*. To view a list of the warnings, errors, and suggestions that are returned by IAM Access Analyzer, see IAM Access Analyzer policy check reference.

When you use the API, the request to create a Multi-Region Access Point is asynchronous. When you submit a request to create a Multi-Region Access Point, Amazon S3 synchronously authorizes the request. It then immediately returns a token that you can use to track the progress of the creation request. For more information about tracking asynchronous requests to create and manage Multi-Region Access Points, see Using Multi-Region Access Points with supported API operations.

After you create the Multi-Region Access Point, you can create an access control policy for it. Each Multi-Region Access Point can have an associated policy. A Multi-Region Access Point policy is a resource-based policy that you can use to limit the use of the Multi-Region Access Point by resource, user, or other conditions.

Note

For an application or user to be able to access an object through a Multi-Region Access Point, both of the following policies must permit the request:

- The access policy for the Multi-Region Access Point
- The access policy for the underlying bucket that contains the object

When the two policies are different, the more restrictive policy takes precedence. To simplify permissions management for Multi-Region Access Points, you can delegate access control from the bucket to the Multi-Region Access Point. For more information, see the section called “Multi-Region Access Point policy examples”.

Using a bucket with a Multi-Region Access Point doesn't change the bucket's behavior when the bucket is accessed through the existing bucket name or an Amazon Resource Name (ARN). All existing operations against the bucket continue to work as before. Restrictions that you include in a Multi-Region Access Point policy apply only to requests that are made through the Multi-Region Access Point.

You can update the policy for a Multi-Region Access Point after creating it, but you can't delete the policy. However, you can update the Multi-Region Access Point policy to deny all permissions.

Topics

- Rules for naming Amazon S3 Multi-Region Access Points
- Rules for choosing buckets for Amazon S3 Multi-Region Access Points
- Create an Amazon S3 Multi-Region Access Point
- Blocking public access with Amazon S3 Multi-Region Access Points
- Viewing Amazon S3 Multi-Region Access Points configuration details
- Deleting a Multi-Region Access Point

Rules for naming Amazon S3 Multi-Region Access Points

When you create a Multi-Region Access Point, you give it a name, which is a string that you choose. You can't change the name of the Multi-Region Access Point after it is created. The name must be unique in your AWS account, and it must conform to the naming requirements listed in Multi-Region Access Point restrictions and limitations. To help you identify the Multi-Region Access Point, use a name that is meaningful to you, to your organization, or that reflects the scenario.

You use this name when invoking Multi-Region Access Point management operations, such as `GetMultiRegionAccessPoint` and `PutMultiRegionAccessPointPolicy`. The name is not used to send requests to the Multi-Region Access Point, and it doesn't need to be exposed to clients who make requests by using the Multi-Region Access Point.

When Amazon S3 creates a Multi-Region Access Point, it automatically assigns an alias to it. This alias is a unique alphanumeric string that ends in `.mrap`. The alias is used to construct the hostname and the Amazon Resource Name (ARN) for a Multi-Region Access Point. The fully qualified name is also based on the alias for the Multi-Region Access Point.

You can't determine the name of a Multi-Region Access Point from its alias, so you can disclose an alias without risk of exposing the name, purpose, or owner of the Multi-Region Access Point. Amazon S3 selects the alias for each new Multi-Region Access Point, and the alias can't be changed. For more information about addressing a Multi-Region Access Point, see Making requests through a Multi-Region Access Point.

Multi-Region Access Point aliases are unique throughout time and aren't based on the name or configuration of a Multi-Region Access Point. If you create a Multi-Region Access Point, and then

delete it and create another one with the same name and configuration, the second Multi-Region Access Point will have a different alias than the first. New Multi-Region Access Points can never have the same alias as a previous Multi-Region Access Point.

Rules for choosing buckets for Amazon S3 Multi-Region Access Points

Each Multi-Region Access Point is associated with the Regions where you want to fulfill requests. The Multi-Region Access Point must be associated with exactly one bucket in each of those Regions. You specify the name of each bucket in the request to create the Multi-Region Access Point. Buckets that support the Multi-Region Access Point can either be in the same AWS account that owns the Multi-Region Access Point, or they can be in other AWS accounts.

A single bucket can be used by multiple Multi-Region Access Points.

Important

- You can specify the buckets that are associated with a Multi-Region Access Point only at the time that you create it. After it is created, you can’t add, modify, or remove buckets from the Multi-Region Access Point configuration. To change the buckets, you must delete the entire Multi-Region Access Point and create a new one.
- You can't delete a bucket that is part of a Multi-Region Access Point. If you want to delete a bucket that's attached to a Multi-Region Access Point, delete the Multi-Region Access Point first.
- If you add a bucket that's owned by another account to your Multi-Region Access Point, the bucket owner must also update their bucket policy to grant access permissions to the Multi-Region Access Point. Otherwise, the Multi-Region Access Point won't be able to retrieve data from that bucket. For example policies that show how to grant such access, see Multi-Region Access Point policy examples.
- Not all Regions support Multi-Region Access Points. To see the list of supported Regions, see Multi-Region Access Point restrictions and limitations.

You can create replication rules to synchronize data between buckets. These rules enable you to automatically copy data from source buckets to destination buckets. Having buckets connected to a Multi-Region Access Point does not affect how replication works. Configuring replication with Multi-Region Access Points is described in a later section.

Important

When you make a request to a Multi-Region Access Point, the Multi-Region Access Point isn't aware of the data contents of the buckets in the Multi-Region Access Point. Therefore, the bucket that gets the request might not contain the requested data. To create consistent datasets in the Amazon S3 buckets that are associated with a Multi-Region Access Point, we recommend that you configure S3 Cross-Region Replication (CRR). For more information, see Configuring replication for use with Multi-Region Access Points.

Create an Amazon S3 Multi-Region Access Point

The following example demonstrates how to create a Multi-Region Access Point by using the Amazon S3 console.

Using the S3 console

To create a Multi-Region Access Point

Note

Multi-Region Access Point opt-in Regions aren't currently supported in the Amazon S3 console.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Choose **Create Multi-Region Access Points** to begin creating your Multi-Region Access Point.
4. On the **Multi-Region Access Point** page, supply a name for the Multi-Region Access Point in the **Multi-Region Access Point name** field.
5. Select the buckets that will be associated with this Multi-Region Access Point. You can choose buckets that are in your account, or you can choose buckets from other accounts.

   Note

   You must add at least one bucket from either your account or other accounts. Also, be aware that Multi-Region Access Points support only one bucket per AWS Region.

Therefore, you can't add two buckets from the same Region. AWS Regions that are disabled by default are not supported.

- To add a bucket that is in your account, choose **Add buckets**. A list of all the buckets in your account displays. You can search for your bucket by name, or sort the bucket names in alphabetical order.
- To add a bucket from another account, choose **Add bucket from other accounts**. Make sure that you know the exact bucket name and AWS account ID because you can't search or browse for buckets in other accounts.

> **Note**
>
> You must enter a valid AWS account ID and bucket name. The bucket must also be in a supported Region, or you will encounter an error when you try to create your Multi-Region Access Point. For the list of Regions that support Multi-Region Access Points, see Multi-Region Access Points restrictions and limitations.

6. (Optional) If you need to remove a bucket that you added, choose **Remove**.

> **Note**
>
> You can't add or remove buckets to this Multi-Region Access Point after you've finished creating it.

7. Under **Block Public Access settings for this Multi-Region Access Point**, select the Block Public Access settings that you want to apply to the Multi-Region Access Point. By default, all Block Public Access settings are enabled for new Multi-Region Access Points. We recommend that you leave all settings enabled unless you know that you have a specific need to disable any of them.

> **Note**
>
> You can't change the Block Public Access settings for a Multi-Region Access Point after the Multi-Region Access Point has been created. Therefore, if you're going to block public access, make sure that your applications work correctly without public access before you create a Multi-Region Access Point.

8. Choose **Create Multi-Region Access Point**.

> **Important**
>
> When you add a bucket that's owned by another account to your Multi-Region Access Point, the bucket owner must also update their bucket policy to grant access permissions to the Multi-Region Access Point. Otherwise, the Multi-Region Access Point won't be able to retrieve data from that bucket. For example policies that show how to grant such access, see Multi-Region Access Point policy examples.

Using the AWS CLI

You can use the AWS CLI to create a Multi-Region Access Point. When you create the Multi-Region Access Point, you must provide all the buckets that it will support. You can't add buckets to the Multi-Region Access Point after it has been created.

The following example creates a Multi-Region Access Point with two buckets by using the AWS CLI. To use this example command, replace the *`user input placeholders`* with your own information.

> **Note**
>
> To create a Multi-Region Access Point using buckets in an opt-in Region, make sure to enable all opt-in Regions first. Otherwise, you'll get a `403 InvalidRegion` error when you try to create a Multi-Region Access Point using buckets for an opt-in Region that you haven't actually opted in to.

```
aws s3control create-multi-region-access-point --account-id 111122223333 --details '{
        "Name": "simple-multiregionaccesspoint-with-two-regions",
        "PublicAccessBlock": {
            "BlockPublicAcls": true,
            "IgnorePublicAcls": true,
            "BlockPublicPolicy": true,
            "RestrictPublicBuckets": true
        },
        "Regions": [
            { "Bucket": "amzn-s3-demo-bucket1" },
```

```
            { "Bucket": "amzn-s3-demo-bucket2" }
        ]
    }' --region us-west-2
```

Blocking public access with Amazon S3 Multi-Region Access Points

Each Multi-Region Access Point has distinct settings for Amazon S3 Block Public Access. These settings operate in conjunction with the Block Public Access settings for the AWS account that owns the Multi-Region Access Point and the underlying buckets.

When Amazon S3 authorizes a request, it applies the most restrictive combination of these settings. If the Block Public Access settings for any of these resources (the Multi-Region Access Point owner account, the underlying bucket, or the bucket owner account) block access for the requested action or resource, Amazon S3 rejects the request.

We recommend that you enable all Block Public Access settings unless you have a specific need to disable any of them. By default, all Block Public Access settings are enabled for a Multi-Region Access Point. If Block Public Access is enabled, the Multi-Region Access Point can't accept internet-based requests.

> **Important**
>
> You can't change the Block Public Access settings for a Multi-Region Access Point after it has been created.

For more information about Amazon S3 Block Public Access, see Blocking public access to your Amazon S3 storage.

Viewing Amazon S3 Multi-Region Access Points configuration details

The following example demonstrates how to view Multi-Region Access Point configuration details by using the Amazon S3 console.

Using the S3 console

To create a Multi-Region Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.

3. Choose the name of the Multi-Region Access Point for which you want to view the configuration details.

   - The **Properties** tab lists all of the buckets that are associated with your Multi-Region Access Point, the creation date, the Amazon Resource Name (ARN), and the alias. The AWS account ID column also lists any buckets owned by external accounts that are associated with your Multi-Region Access Point.
   - The **Permissions** tab lists the Block Public Access settings that are applied to the buckets associated with this Multi-Region Access Point. You can also view the Multi-Region Access Point policy for your Multi-Region Access Point, if you've created one. The **Info** alert on the **Permissions** page also lists all the buckets (in your account and other accounts) for this Multi-Region Access Point that have the **Public Access is blocked** setting enabled.
   - The **Replication and failover** tab provides a map view of the buckets that are associated with your Multi-Region Access Point and the Regions that the buckets reside int. If there are buckets from another account that you don't have permission to pull data from, the Region will be marked in red on the **Replication summary** map, indicating that it is an **AWS Region with errors getting replication status**.

     > **Note**
     >
     > To retrieve replication status information from a bucket in an external account, the bucket owner must grant you the `s3:GetBucketReplication` permission in their bucket policy.

     This tab also provides the replication metrics, replication rules, and failover statuses for the Regions that are used with your Multi-Region Access Point.

Using the AWS CLI

You can use the AWS CLI to view the configuration details for a Multi-Region Access Point.

The following AWS CLI example gets your current Multi-Region Access Point configuration. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control get-multi-region-access-point --account-id 111122223333 --name amzn-s3-demo-bucket
```

Deleting a Multi-Region Access Point

The following procedure explains how to delete a Multi-Region Access Point by using the Amazon S3 console. Be aware that deleting a Multi-Region Access Point doesn't delete the buckets associated with the Multi-Region Access Point. Instead, it only deletes the Multi-Region Access Point itself.

> **Note**
>
> S3 Multi-Region Access Points using buckets in AWS opt-in Regions is currently only supported through AWS SDKs and the AWS CLI. To delete a Multi-Region Access Point using buckets in an opt-in Region, make sure to specify which AWS opt-in Regions your account can use first. Otherwise, if you try to delete a Multi-Region Access Point that uses buckets in disabled AWS opt-in Regions, you'll receive an error.

Using the S3 console

To delete a Multi-Region Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Select the option button next to the name of your Multi-Region Access Point.
4. Choose **Delete**.
5. In the **Delete Multi-Region Access Point** dialog box, enter the name of the AWS bucket that you want to delete.

   > **Note**
   >
   > Make sure to enter a valid bucket name. Otherwise, the **Delete** button will be disabled.

6. Choose **Delete** to confirm deletion of your Multi-Region Access Point.

Using the AWS CLI

You can use the AWS CLI to delete a Multi-Region Access Point. This action does not delete the buckets associated with the Multi-Region Access Point, only the Multi-Region Access Point itself. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control delete-multi-region-access-point --account-id 123456789012 --details
 Name=example-multi-region-access-point-name
```

Configuring a Multi-Region Access Point for use with AWS PrivateLink

You can use Multi-Region Access Points to route Amazon S3 request traffic between AWS Regions. Each Multi-Region Access Point global endpoint routes Amazon S3 data request traffic from multiple sources without your having to build complex networking configurations with separate endpoints. These data-request traffic sources include:

- Traffic originating in a virtual private cloud (VPC)
- Traffic from on-premises data centers traveling over AWS PrivateLink
- Traffic from the public internet

If you establish an AWS PrivateLink connection to an S3 Multi-Region Access Point, you can route S3 requests into AWS, or across multiple AWS Regions, over a private connection by using a simple network architecture and configuration. When you use AWS PrivateLink, you don't need to configure a VPC peering connection.

Topics

- Configuring Multi-Region Access Point opt-in Regions
- Configuring a Multi-Region Access Point for use with AWS PrivateLink
- Removing access to a Multi-Region Access Point from a VPC endpoint

Configuring Multi-Region Access Point opt-in Regions

An AWS opt-in Region is a Region that isn't enabled by default in your AWS account. In contrast, Regions that are enabled by default are known as AWS Regions or commercial Regions.

To start using Multi-Region Access Points in AWS opt-in Regions, you must manually enable the opt-in Region for your AWS account before creating your Multi-Region Access Point. After you

enable the opt-in Region, you can create Multi-Region Access Points with buckets in the selected opt-in Region. For instructions on how to enable or disable an opt-in Region for your AWS account or AWS Organization, see Enable or disable a Region for standalone accounts or Enable or disable a Region in your organization.

> **Note**
>
> Multi-Region Access Point opt-in Regions are currently only supported through AWS SDKs and AWS CLI.

S3 Multi-Region Access Points supports the following AWS opt-in Regions:

- `Africa (Cape Town)`
- `Asia Pacific (Hong Kong)`
- `Asia Pacific (Jakarta)`
- `Asia Pacific (Melbourne)`
- `Asia Pacific (Hyderabad)`
- `Canada West (Calgary)`
- `Europe (Zurich)`
- `Europe (Milan)`
- `Europe (Spain)`
- `Israel (Tel Aviv)`
- `Middle East (Bahrain)`
- `Middle East (UAE)`

> **Note**
>
> There are no additional costs for enabling an opt-in Region. However, creating or using a resource in a Multi-Region Access Point results in billing charges.

Using a Multi-Region Access Point in an AWS opt-in Region

To perform a data plane operation on your Multi-Region Access Point, all associated AWS accounts must enable the opt-in Regions that are part of the Multi-Region Access Point. This requirement applies to the requester account, the Multi-Region Access Point owner, S3 bucket owners, and the VPC endpoint owner. If any of these accounts don't enable AWS opt-in Regions, the Multi-Region Access Point requests fail. For more information about the `InvalidToken` or `AllAccessDisabled` errors, see List of error codes.

> **Note**
>
> Control plane operations such as updating your Multi-Region Access Point policy or updating your failover configuration aren't impacted by the opt-in Region status of any Region that is part of your Multi-Region Access Point. You also don't need to disable any active opt-in Regions before deleting a Multi-Region Access Point.

Disabling an active AWS opt-in Region

If you disable opt-in Region that is part of your Multi-Region Access Point, requests routed to this Region result in a `403 AllAccessDisabled` error. To safely disable an opt-in Region, we recommend that you first identify an alternate Region in your Multi-Region Access Point configuration to route the traffic to. You can then use Multi-Region Access Point failover controls to mark the alternate Region as active, and mark the Region to be disabled as passive. After changing the failover controls, you can disable the Region you want to opt out of.

Enabling a previously disabled AWS opt-in Region

To enable an opt-in AWS Region that was previously disabled for your Multi-Region Access Point, make sure to update your AWS account settings. After you re-enable the opt-in Region, run the PutMultiRegionAccessPointPolicy API operation to apply the Multi-Region Access Points policy to the opt-in Region.

If your Multi-Region Access Point is accessed through a VPC endpoint, we recommend that you update your VPCE policy and use the ModifyVpcEndpoint API operation to apply the updated VPC endpoint policy to the re-enabled opt-in Region.

Multi-Region Access Points policy and multiple AWS accounts

If your Multi-Region Access Points policy grants access to multiple AWS accounts, all requester accounts must also enable the same opt-in Regions in their account settings. If the requester account submits a Multi-Region Access Point request without enabling the opt-in Regions that are part of the Multi-Region Access Point, it'll result in a `400 InvalidToken` error.

AWS opt-in Region considerations

When you access a Multi-Region Access Point from an opt-in Region, be aware of the following:

- When you enable an opt-in Region, it allows you to create a Multi-Region Access Point using the buckets from the opt-in Region. When you disable an opt-in Region, the Multi-Region Access Point is no longer supported in the opt-in Region. If you no longer want an opt-in Region enabled for your Multi-Region Access Point, make sure to disable the Region for your account first. Then, create a new Multi-Region Access Point with your preferred list of opt-in Regions.
- If you attempt to create your Multi-Region Access Point with a disabled opt-in Region, you'll receive a `403 InvalidRegion` error. After you enable the opt-in Region, try creating the Multi-Region Access Point again.
- The maximum number of supported Regions for a Multi-Region Access Point is 17 Regions. This includes both opt-in Regions and commercial Regions. For more information, see Multi-Region Access Points restrictions and limitations.
- Control plane requests for Multi-Region Access Points will work, even if you haven't opted in to any Regions.
- When you're trying to create a Multi-Region Access Point for the first time, you must opt into all Regions that are part of the Multi-Region Access Point.
- Any AWS accounts that are granted access to an S3 Multi-Region Access Point through the Multi-Region Access Point policy must also enable the same opt-in Regions that are part of the Multi-Region Access Point.

Configuring a Multi-Region Access Point for use with AWS PrivateLink

AWS PrivateLink provides you with private connectivity to Amazon S3 using private IP addresses in your virtual private cloud (VPC). You can provision one or more interface endpoints inside your VPC to connect to Amazon S3 Multi-Region Access Points.

You can create **com.amazonaws.s3-global.accesspoint** endpoints for Multi-Region Access Points through the AWS Management Console, AWS CLI, or AWS SDKs. To learn more about how to

configure an interface endpoint for Multi-Region Access Point, see Interface VPC endpoints in the *VPC User Guide*.

To make requests to a Multi-Region Access Point through interface endpoints, follow these steps to configure the VPC and the Multi-Region Access Point.

To configure a Multi-Region Access Point to use with AWS PrivateLink

1. Create or have an appropriate VPC endpoint that can connect to Multi-Region Access Points. For more information about creating VPC endpoints, see Interface VPC endpoints in the *VPC User Guide*.

   > **Important**
   >
   > Make sure to create a **com.amazonaws.s3-global.accesspoint** endpoint. Other endpoint types cannot access Multi-Region Access Points.

   After this VPC endpoint is created, all Multi-Region Access Point requests in the VPC route through this endpoint if you have private DNS enabled for the endpoint. This is enabled by default.
2. If the Multi-Region Access Point policy does not support connections from VPC endpoints, you will need to update it.
3. Verify that the individual bucket policies will allow access to the users of the Multi-Region Access Point.

Remember that Multi-Region Access Points work by routing requests to buckets, not by fulfilling requests themselves. This is important to remember because the originator of the request must have permissions to the Multi-Region Access Point and be allowed to access the individual buckets in the Multi-Region Access Point. Otherwise, the request might be routed to a bucket where the originator doesn't have permissions to fulfill the request. A Multi-Region Access Point and the buckets associated can be owned by the same or another AWS account. However, VPCs from different accounts can use a Multi-Region Access Point if the permissions are configured correctly.

Because of this, the VPC endpoint policy must allow access both to the Multi-Region Access Point and to each underlying bucket that you want to be able to fulfill requests.

As mentioned previously, you also must make sure that the Multi-Region Access Point policy is configured to support access through a VPC endpoint. You don't need to specify the VPC endpoint that is requesting access. The following sample policy would grant access to any requester trying to use the Multi-Region Access Point for the `GetObject` requests.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
    {
        "Sid": "Open-read-MRAP-policy",
        "Effect": "Allow",
        "Principal": "*",
        "Action": [
            "s3:GetObject"
          ],
        "Resource": "arn:aws:s3::111122223333:accesspoint/mfzwi23gnjvgw.mrap/
object/*"
    }]
}
```

And of course, the individual buckets would each need a policy to support access from requests submitted through VPC endpoint. The following example policy grants read access to any anonymous users. This would include requests made through the VPC endpoint.

JSON

```
{
    "Version":"2012-10-17",
   "Statement": [
   {
       "Sid": "Public-read",
       "Effect": "Allow",
       "Principal": "*",
       "Action": "s3:GetObject",
       "Resource": [
           "arn:aws:s3:::amzn-s3-demo-bucket1",
           "arn:aws:s3:::amzn-s3-demo-bucket2/*"]
    }]
```

```
}
```

For more information about editing a VPC endpoint policy, see Control access to services with VPC endpoints in the *VPC User Guide*.

Removing access to a Multi-Region Access Point from a VPC endpoint

If you own a Multi-Region Access Point and want to remove access to it from an interface endpoint, you must supply a new access policy for the Multi-Region Access Point that prevents access for requests coming through VPC endpoints. However, if the buckets in your Multi-Region Access Point support requests through VPC endpoints, they will continue to support these requests. If you want to prevent that support, you must also update the policies for the buckets. Supplying a new access policy to the Multi-Region Access Point prevents access only to the Multi-Region Access Point, not to the underlying buckets.

> **Note**
>
> You can't delete an access policy for a Multi-Region Access Point. To remove access to a Multi-Region Access Point, you must provide a new access policy with the modified access that you want.

Instead of updating the access policy for the Multi-Region Access Point, you can update the bucket policies to prevent requests through VPC endpoints. In this case, users can still access the Multi-Region Access Point through the VPC endpoint. However, if the Multi-Region Access Point request is routed to a bucket where the bucket policy prevents access, the request will generate an error message.

Making requests through a Multi-Region Access Point

Like other resources, Amazon S3 Multi-Region Access Points have Amazon Resource Names (ARNs). You can use these ARNs to direct requests to Multi-Region Access Points by using the AWS Command Line Interface (AWS CLI), AWS SDKs, or the Amazon S3 API. You can also use these ARNs to identify Multi-Region Access Points in access control policies. A Multi-Region Access Point ARN doesn't include or disclose the name of the Multi-Region Access Point. For more information about ARNs, see Amazon Resource Names (ARNs) in the *AWS General Reference*.

> **Note**
>
> The Multi-Region Access Point alias and ARN cannot be used interchangeably.

Multi-Region Access Point ARNs use the following format:

`arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

The following are a few examples of Multi-Region Access Point ARNs:

- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap` represents the Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, which is owned by AWS account `123456789012`.
- `arn:aws:s3::123456789012:accesspoint/*` represents all Multi-Region Access Points under the account `123456789012`. This ARN matches all Multi-Region Access Points for account `123456789012`, but doesn't match any Regional Amazon S3 Access Points because the ARN doesn't include an AWS Region. In contrast, the ARN `arn:aws:s3:us-west-2:123456789012:accesspoint/*` matches all Regional Amazon S3 Access Points in the Region `us-west-2` for the account `123456789012`, but doesn't match any Multi-Region Access Points.

ARNs for objects that are accessed through a Multi-Region Access Point use the following format:

`arn:aws:s3::account_id:accesspoint/MultiRegionAccessPoint_alias//key`

As with Multi-Region Access Point ARNs, the ARNs for objects that are accessed through Multi-Region Access Points don't include an AWS Region. Here are some examples.

- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01` represents the object with key `-01`, which is accessed through the Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, which is owned by account `123456789012`.
- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//*` represents all objects that can be accessed through the Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, in account `123456789012`.
- `arn:aws:s3::123456789012:accesspoint/mfzwi23gnjvgw.mrap//-01/finance/*` represents all objects that can be accessed under the `-01/finance/` for the Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, in account `123456789012`.

Multi-Region Access Point hostnames

You can access data in Amazon S3 through a Multi-Region Access Point by using the hostname of the Multi-Region Access Point. Requests can be directed to this hostname from the public internet. If you have configured one or more internet gateways for the Multi-Region Access Point, requests can also be directed to this hostname from a virtual private cloud (VPC). For more information about creating VPC interface endpoints to use with Multi-Region Access Points, see Configuring a Multi-Region Access Point for use with AWS PrivateLink.

To make requests through a Multi-Region Access Point from a VPC by using a VPC endpoint, you can use AWS PrivateLink. When you're making requests to a Multi-Region Access Point by using AWS PrivateLink, you cannot directly use an endpoint-specific Regional DOMAIN NAME SYSTEM (DNS) name that ends with *region*`.vpce.amazonaws.com`. This hostname will not have a certificate associated with it, so it cannot be used directly. You can still use the public DOMAIN NAME SYSTEM (DNS) name of the VPC endpoint as a `CNAME` or `ALIAS` target. Alternatively, you can enable private DOMAIN NAME SYSTEM (DNS) on the endpoint and use the standard Multi-Region Access Point *MultiRegionAccessPoint_alias*`.accesspoint.s3-global.amazonaws.com` DOMAIN NAME SYSTEM (DNS) name, as described in this section.

When you make requests to the API for Amazon S3 data operations (for example, `GetObject`) through a Multi-Region Access Point, the hostname for the request is as follows:

MultiRegionAccessPoint_alias`.accesspoint.s3-global.amazonaws.com`

For example, to make a `GetObject` request through the Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, make a request to the hostname `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`. The `s3-global` portion of the hostname indicates that this hostname is not for a specific Region.

Making requests through a Multi-Region Access Point is similar to making requests through a single-Region access point. However, it's important to be aware of the following differences:

- Multi-Region Access Point ARNs don't include an AWS Region. They follow the format `arn:aws:s3::`*account-id*`:accesspoint/`*MultiRegionAccessPoint_alias*.
- For requests made through API operations (these requests don't require the use of an ARN), Multi-Region Access Points use a different endpoint scheme. The scheme is *MultiRegionAccessPoint_alias*`.accesspoint.s3-global.amazonaws.com`—for example, `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`. Note the differences compared to a single-Region access point:

- Multi-Region Access Point hostnames use their alias, not the Multi-Region Access Point name.
- Multi-Region Access Point hostnames don't include the owner's AWS account ID.
- Multi-Region Access Point hostnames don't include an AWS Region.
- Multi-Region Access Point hostnames include `s3-global.amazonaws.com` instead of `s3.amazonaws.com`.

- Multi-Region Access Point requests must be signed by using Signature Version 4A (SigV4A). When you use the AWS SDKs, the SDK automatically converts a SigV4 to SigV4A. Therefore, make sure that your AWS SDK supports SigV4A as the signing implementation that is used to sign the global AWS Region requests. For more information about SigV4A, see Signing AWS API requests in the *AWS General Reference*.

Multi-Region Access Points and Amazon S3 Transfer Acceleration

Amazon S3 Transfer Acceleration is a feature that enables fast transfer of data to buckets. Transfer Acceleration is configured on the individual bucket level. For more information about Transfer Acceleration, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

Multi-Region Access Points use a similar accelerated transfer mechanism as Transfer Acceleration for sending large objects over the AWS network. Because of this, you don't need to use Transfer Acceleration when sending requests through a Multi-Region Access Point. This increased transfer performance is automatically incorporated into the Multi-Region Access Point.

Topics

- Permissions
- Multi-Region Access Point restrictions and limitations
- Multi-Region Access Point request routing
- Amazon S3 Multi-Region Access Points failover controls
- Configuring replication for use with Multi-Region Access Points
- Using Multi-Region Access Points with supported API operations
- Monitoring and logging requests made through a Multi-Region Access Point to underlying resources

Permissions

Amazon S3 Multi-Region Access Points can simplify data access for Amazon S3 buckets in multiple AWS Regions. Multi-Region Access Points are named global endpoints that you can use to perform Amazon S3 data-access object operations, such as `GetObject` and `PutObject`. Each Multi-Region Access Point can have distinct permissions and network controls for any request that is made through the global endpoint.

Each Multi-Region Access Point can also enforce a customized access policy that works in conjunction with the bucket policy that is attached to the underlying bucket. For a cross-account request to succeed, the following policies must permit the operation:

- The Multi-Region Access Point policy
- The underlying AWS Identity and Access Management (IAM) policy
- The underlying bucket policy (where the request is routed to)

> **Note**
>
> For same account requests, only the underlying IAM policy, which grants the appropriate access, is required.

You can configure any Multi-Region Access Point policy to accept requests only from specific IAM users or groups. For an example of how to do this, see Example 2 in the section called "Multi-Region Access Point policy examples". To restrict Amazon S3 data access to a private network, you can configure the Multi-Region Access Point policy to accept requests only from a virtual private cloud (VPC).

For example, suppose that you make a `GetObject` request through a Multi-Region Access Point by using a user called `AppDataReader` in your AWS account. To help ensure that the request won't be denied, the `AppDataReader` user must be granted the `s3:GetObject` permission by the Multi-Region Access Point and by each bucket underlying the Multi-Region Access Point. `AppDataReader` won't be able to retrieve data from any bucket that doesn't grant this permission.

> **Important**
>
> Delegating access control for a bucket to a Multi-Region Access Point policy doesn't change the bucket's behavior when the bucket is accessed directly through its bucket name

or Amazon Resource Name (ARN). All operations made directly against the bucket will continue to work as before. Restrictions that you include in a Multi-Region Access Point policy apply only to requests made through that Multi-Region Access Point.

Managing public access to a Multi-Region Access Point

Multi-Region Access Points support independent Block Public Access settings for each Multi-Region Access Point. When you create a Multi-Region Access Point, you can specify the Block Public Access settings that apply to that Multi-Region Access Point.

Note

Any Block Public Access settings that are enabled under **Block Public Access settings for this account** (in your own account) or **Block Public Settings for external buckets** still apply even if the independent Block Public Access settings for your Multi-Region Access Point are disabled.

For any request that is made through a Multi-Region Access Point, Amazon S3 evaluates the Block Public Access settings for:

- The Multi-Region Access Point
- The underlying buckets (including external buckets)
- The account that owns the Multi-Region Access Point
- The account that owns the underlying buckets (including external accounts)

If any of these settings indicate that the request should be blocked, Amazon S3 rejects the request. For more information about the Amazon S3 Block Public Access feature, see Blocking public access to your Amazon S3 storage.

Important

By default, all Block Public Access settings are enabled for Multi-Region Access Points. You must explicitly turn off any settings that you don't want to apply to a Multi-Region Access Point.

You can't change the Block Public Access settings for a Multi-Region Access Point after it has been created.

Viewing Block Public Access settings for a Multi-Region Access Point

To view the Block Public Access settings for a Multi-Region Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Choose the name of the Multi-Region Access Point that you want to review.
4. Choose the **Permissions** tab.
5. Under **Block Public Access settings for this Multi-Region Access Point**, review the Block Public Access settings for your Multi-Region Access Point.

> **Note**
>
> You can't edit the Block Public Access settings after the Multi-Region Access Point is created. Therefore, if you're going to block public access, make sure that your applications work correctly without public access before you create a Multi-Region Access Point.

Using a Multi-Region Access Point policy

The following example Multi-Region Access Point policy grants an IAM user access to list and download files from your Multi-Region Access Point. To use this example policy, replace the *`user input placeholders`* with your own information.

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
     {
        "Effect":"Allow",
```

```
         "Principal":{
            "AWS":"arn:aws:iam::123456789012:user/JohnDoe"
         },
         "Action":[
            "s3:ListBucket",
            "s3:GetObject"
         ],
         "Resource":[
            "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
            "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/
object/*"
         ]
      }
   ]
}
```

To associate your Multi-Region Access Point policy with the specified Multi-Region Access Point by using the AWS Command Line Interface (AWS CLI), use the following `put-multi-region-access-point-policy` command. To use this example command, replace the *`user input placeholders`* with your own information. Each Multi-Region Access Point can have only one policy, so a request made to the `put-multi-region-access-point-policy` action replaces any existing policy that is associated with the specified Multi-Region Access Point.

AWS CLI

```
aws s3control put-multi-region-access-point-policy
--account-id 111122223333
--details { "Name": "amzn-s3-demo-bucket-MultiRegionAccessPoint",
 "Policy": "{ \"Version\": \"2012-10-17\", \"Statement\": { \"Effect\":
 \"Allow\", \"Principal\": { \"AWS\": \"arn:aws:iam::111122223333:root
\" }, \"Action\": [\"s3:ListBucket\", \"s3:GetObject\"], \"Resource\":
 [ \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
 \"arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/object/*
\" ] } }" }
```

To query your results for the previous operation, use the following command:

AWS CLI

```
aws s3control describe-multi-region-access-point-operation
```

```
--account-id 111122223333
--request-token-arn requestArn
```

To retrieve your Multi-Region Access Point policy, use the following command:

AWS CLI

```
aws s3control get-multi-region-access-point-policy
--account-id 111122223333
--name=amzn-s3-demo-bucket-MultiRegionAccessPoint
```

Editing the Multi-Region Access Point policy

The Multi-Region Access Point policy (written in JSON) provides storage access to the Amazon S3 buckets that are used with this Multi-Region Access Point. You can allow or deny specific principals to perform various actions on your Multi-Region Access Point. When a request is routed to a bucket through the Multi-Region Access Point, both the access policies for the Multi-Region Access Point and the bucket apply. The more restrictive access policy always takes precedence.

> **Note**
>
> If a bucket contains objects that are owned by other accounts, the Multi-Region Access Point policy doesn't apply to the objects that are owned by other AWS accounts.

After you apply a Multi-Region Access Point policy, the policy cannot be deleted. You can either edit the policy or create a new policy that overwrites the existing one.

To edit the Multi-Region Access Point policy

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Choose the name of the Multi-Region Access Point that you want to edit the policy for.
4. Choose the **Permissions** tab.
5. Scroll down to the **Multi-Region Access Point policy** section. Choose **Edit** to update the policy (in JSON).

6. The **Edit Multi-Region Access Point policy** page appears. You can either enter the policy directly into the text field, or you can choose **Add statement** to select policy elements from a dropdown list.

> **Note**
>
> The console automatically displays the Multi-Region Access Point Amazon Resource Name (ARN), which you can use in the policy. For example Multi-Region Access Point policies, see the section called “Multi-Region Access Point policy examples”.

Multi-Region Access Point policy examples

Amazon S3 Multi-Region Access Points support AWS Identity and Access Management (IAM) resource policies. You can use these policies to control the use of the Multi-Region Access Point by resource, user, or other conditions. For an application or user to be able to access objects through a Multi-Region Access Point, both the Multi-Region Access Point and the underlying bucket must allow the same access.

To allow the same access to both the Multi-Region Access Point and the underlying bucket, do one of the following:

- **(Recommended)** To simplify access controls when using an Amazon S3 Multi-Region Access Point, delegate access control for the Amazon S3 bucket to the Multi-Region Access Point. For an example of how to do this, see Example 1 in this section.
- Add the same permissions contained in the Multi-Region Access Point policy to the underlying bucket policy.

> **Important**
>
> Delegating access control for a bucket to a Multi-Region Access Point policy doesn't change the bucket's behavior when the bucket is accessed directly through its bucket name or Amazon Resource Name (ARN). All operations made directly against the bucket will continue to work as before. Restrictions that you include in a Multi-Region Access Point policy apply only to requests made through that Multi-Region Access Point.

Example 1 – Delegating access to specific Multi-Region Access Points in your bucket policy (for the same account or cross-account)

The following example bucket policy grants full bucket access to a specific Multi-Region Access Point. This means that all access to this bucket is controlled by the policies that are attached to the Multi-Region Access Point. We recommend configuring your buckets this way for all use cases that don't require direct access to the bucket. You can use this bucket policy structure for Multi-Region Access Points in either the same account or in another account.

JSON

```
{
    "Version":"2012-10-17",
    "Statement" : [
    {
        "Effect": "Allow",
        "Principal" : { "AWS": "*" },
        "Action" : "*",
        "Resource" : [ "arn:aws:s3:::amzn-s3-demo-bucket", "arn:aws:s3:::amzn-s3-
demo-bucket/*"],
        "Condition": {
            "StringEquals" : { "s3:DataAccessPointArn" :
 "arn:aws:s3::111122223333:accesspoint/example-multi-region-access-point" }
        }
    }]
}
```

> **Note**
>
> If there are multiple Multi-Region Access Points that you're granting access to, make sure to list each Multi-Region Access Point.

Example 2 – Granting an account access to a Multi-Region Access Point in your Multi-Region Access Point policy

The following Multi-Region Access Point policy allows account *123456789012* permission to list and read the objects contained in the Multi-Region Access Point defined by the *MultiRegionAccessPoint_ARN*.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
       "Effect": "Allow",
       "Principal": {
          "AWS": "arn:aws:iam::111122223333:user/JohnDoe"
       },
       "Action": [
          "s3:ListBucket",
          "s3:GetObject"
       ],
       "Resource": [
          "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias",
          "arn:aws:s3::111122223333:accesspoint/MultiRegionAccessPoint_alias/
object/*"
       ]
     }
  ]
}
```

Example 3 – Multi-Region Access Point policy that allows bucket listing

The following Multi-Region Access Point policy allows account *123456789012* permission to list the objects contained in the Multi-Region Access Point defined by the *MultiRegionAccessPoint_ARN*.

Multi-Region Access Point restrictions and limitations

Multi-Region Access Points in Amazon S3 have the following restrictions and limitations.

Names and aliases

Multi-Region Access Point names must meet the following requirements:

- Must be unique within a single AWS account.
- Must begin with a number or lowercase letter.
- Must be between 3 and 50 characters long.

- Can't begin or end with a hyphen (-).
- Can't contain underscores (_), uppercase letters, or periods (.).
- Can't be edited after they are created.

Multi-Region Access Point aliases (which are different from a Multi-Region Access Point name), are automatically generated by Amazon S3 and can't be edited or reused. For more information about the difference between Multi-Region Access Point aliases and Multi-Region Access Point names and their respective naming rules, see Rules for naming Amazon S3 Multi-Region Access Points.

Accessing a Multi-Region Access Point

You can't access data through a Multi-Region Access Point by using gateway endpoints. However, you can access data through a Multi-Region Access Point by using interface endpoints. To use AWS PrivateLink, you must create VPC endpoints. For more information, see Configuring a Multi-Region Access Point for use with AWS PrivateLink. However, be aware that IPv6 isn't supported.

To use Multi-Region Access Points with Amazon CloudFront, you must configure the Multi-Region Access Point as a `Custom Origin` distribution type. For more information about various origin types, see Using various origins with CloudFront distributions. For more information about using Multi-Region Access Points with Amazon CloudFront, see Building an active-active, proximity-based application across multiple Regions on the *AWS Storage Blog*.

> **Note**
>
> S3 on Outposts buckets aren't supported.

Signing AWS API requests

To sign an AWS API request, your Multi-Region Access Point must meet the following minimum requirements:

> **Note**
>
> Multi-Region Access Points don't support anonymous requests.

- Support for Transport Layer Security (TLS) version 1.2.

- Support for Signature Version 4 (SigV4A)–This version of SigV4 allows requests to be signed for multiple AWS Regions. This feature is useful in API operations that might result in data access from one of several Regions. When using an AWS SDK, you supply your credentials, and the requests to Multi-Region Access Points will use Signature Version 4A without additional configuration. Make sure to check your AWS SDK compatibility with the SigV4A algorithm. For more information about SigV4A, see Signing AWS API requests in the *AWS General Reference*.

> **Note**
>
> To use SigV4A with temporary security credentials—for example, when using AWS Identity and Access Management (IAM) roles—you can request the temporary credentials from a Regional AWS Security Token Service (AWS STS) endpoint. If you request temporary credentials from the global AWS STS endpoint (`sts.amazonaws.com`), then you must first set the Region compatibility of session tokens for the global endpoint to be valid in all AWS Regions. For more information, see Managing AWS STS in an AWS Region in the *IAM User Guide*.

Amazon S3 API operations

- `CopyObject` is supported as a destination only when using the Multi-Region Access Point ARN.
- The S3 Batch Operations feature isn't supported.

AWS SDKs

Certain AWS SDKs aren't supported. To confirm which AWS SDKs are supported for Multi-Region Access Points, see Compatibility with AWS SDKs.

Service quotas

Be aware of the following service quota limitations:

- There is a maximum of 100 Multi-Region Access Points per account.
- There is a limit of 17 Regions for a single Multi-Region Access Point.

Creating, deleting, or modifying a Multi-Region Access Point

When you create, delete, or modify an Multi-Region Access Point, be aware of the following rules and restrictions:

- After you create a Multi-Region Access Point, you can't add, modify, or remove buckets from the Multi-Region Access Point configuration. To change the buckets, you must delete the entire Multi-Region Access Point and create a new one. If a cross-account bucket in your Multi-Region Access Point is deleted, the only way to reconnect this bucket is to recreate the bucket, using the same name and Region in that account.
- Underlying buckets (in the same account) that are used in a Multi-Region Access Point can be deleted only after a Multi-Region Access Point is deleted.

Region support

Control plane requests

All control plane requests to create or maintain Multi-Region Access Points must be routed to the `US West (Oregon)` Region. For Multi-Region Access Point data plane requests, Regions don't need to be specified.

For the Multi-Region Access Point failover control plane, requests must be routed to one of these five supported Regions:

- `US East (N. Virginia)`
- `US West (Oregon)`
- `Asia Pacific (Sydney)`
- `Asia Pacific (Tokyo)`
- `Europe (Ireland)`

Regions enabled by default

Your Multi-Region Access Point supports buckets in the following default AWS Regions (which are enabled by default in your AWS account):

- `US East (N. Virginia)`
- `US East (Ohio)`

- US West (N. California)
- US West (Oregon)
- Asia Pacific (Mumbai)
- Asia Pacific (Osaka)
- Asia Pacific (Seoul)
- Asia Pacific (Singapore)
- Asia Pacific (Sydney)
- Asia Pacific (Tokyo)
- Canada (Central)
- Europe (Frankfurt)
- Europe (Ireland)
- Europe (London)
- Europe (Paris)
- Europe (Stockholm)
- South America (São Paulo)

AWS opt-in Regions

Your Multi-Region Access Point also supports buckets in the following opt-in AWS Regions (which are disabled by default in your AWS account):

- Africa (Cape Town)
- Asia Pacific (Hong Kong)
- Asia Pacific (Jakarta)
- Asia Pacific (Melbourne)
- Asia Pacific (Hyderabad)
- Canada West (Calgary)
- Europe (Zurich)
- Europe (Milan)
- Europe (Spain)
- Israel (Tel Aviv)

- `Middle East (Bahrain)`
- `Middle East (UAE)`

> **Note**
>
> There are no additional costs for enabling an opt-in Region. However, creating or using a resource in a Multi-Region Access Point results in billing charges.

An opt-in Region must be manually enabled when configuring or creating your Multi-Region Access Point. For more information about opt-in Region behaviors for Multi-Region Access Points, see Configuring Multi-Region Access Point opt-in Regions. For information about how to enable an opt-in Region in your AWS account, see Enable or disable a Region for standalone accounts in the *AWS Account Management Reference Guide*.

Multi-Region Access Point request routing

When you make a request through a Multi-Region Access Point, Amazon S3 determines which of the buckets that are associated with the Multi-Region Access Point is closest to you. Amazon S3 then directs the request to that bucket, regardless of the AWS Region it is located in.

After the Multi-Region Access Point routes the request to the closest-proximity bucket, Amazon S3 processes the request as if you made it directly to that bucket. Multi-Region Access Points aren't aware of the data contents of an Amazon S3 bucket. Therefore, the bucket that gets the request might not contain the requested data. To create consistent datasets in the Amazon S3 buckets that are associated with a Multi-Region Access Point, you can configure S3 Cross-Region Replication (CRR). Then any bucket can fulfill the request successfully.

Amazon S3 directs Multi-Region Access Point requests according to the following rules:

- Amazon S3 optimizes requests to be fulfilled according to proximity. It looks at the buckets supported by the Multi-Region Access Point and relays the request to the bucket that has the closest proximity.
- If the request specifies an existing resource (for example, `GetObject`), Amazon S3 does *not* consider the name of the object when fulfilling the request. This means that even if an object exists in one bucket in the Multi-Region Access Point, your request can be routed to a bucket that doesn't contain the object. This situation will result in a 404 error message being returned to the client.

To avoid 404 errors, we recommend that you configure S3 Cross-Region Replication (CRR) for your buckets. Replication helps resolve the potential issue when the object that you want is in a bucket in the Multi-Region Access Point, but it's not located in the specific bucket that your request was routed to. For more information about configuring replication, see Configuring replication for use with Multi-Region Access Points.

To ensure that your requests are fulfilled by using the specific objects that you want, we also recommend that you turn on bucket versioning and include version IDs in your requests. This approach helps ensure that you have the correct version of the object that you are looking for. Versioning-enabled buckets can also help you recover objects from accidental overwrite. For more information, see Using S3 Versioning in S3 buckets.

- If the request is to create a resource (for example, `PutObject` or `CreateMultipartUpload`), Amazon S3 fulfills the request by using the closest-proximity bucket. For example, consider a video company that wants to support video uploads from anywhere in the world. When a user makes a PUT request to the Multi-Region Access Point, the object is put into the bucket with the closest proximity. To then make that uploaded video available to others around the world for download with the lowest latency, you can use CRR with bidirectional (two-way) replication. Using CRR with two-way replication keeps the contents of all the buckets that are associated with the Multi-Region Access Point synchronized. For more information about using replication with Multi-Region Access Points, see Configuring replication for use with Multi-Region Access Points.

Amazon S3 Multi-Region Access Points failover controls

With Amazon S3 Multi-Region Access Point failover controls, you can maintain business continuity during Regional traffic disruptions, while also giving your applications a multi-Region architecture to fulfill compliance and redundancy needs. If your Regional traffic gets disrupted, you can use Multi-Region Access Point failover controls to select which AWS Regions behind an Amazon S3 Multi-Region Access Point will process data-access and storage requests.

To support failover, you can set up your Multi-Region Access Point in an active-passive configuration, with traffic flowing to the active Region during normal conditions, and a passive Region on standby for failover.

For example, to perform failover to an AWS Region of your choice, you shift traffic from your primary (active) Region to your secondary (passive) Region. In an active-passive configuration like this, one bucket is active and accepting traffic, while the other bucket is passive and not accepting

traffic. The passive bucket is used for disaster recovery. When you initiate failover, all traffic (such as GET or PUT requests) is directed to the bucket in the active state (in one Region) and away from the bucket in the passive state (in another Region).

If you have S3 Cross-Region Replication (CRR) enabled with two-way replication rules, you can keep your buckets synchronized during a failover. In addition, if you have CRR enabled in an active-active configuration, Amazon S3 Multi-Region Access Points can also fetch data from the bucket location of closest proximity, which improves application performance.

AWS Region support

With Amazon S3 Multi-Region Access Points failover controls, your S3 buckets can be in any of the 17 Regions where Multi-Region Access Points are supported. You can initiate failover across any two Regions at one time.

> **Note**
>
> Although failover is initiated between only two Regions at one time, you can separately update the routing statuses for multiple Regions at the same time in your Multi-Region Access Point.

The following topics demonstrate how to use and manage Amazon S3 Multi-Region Access Point failover controls.

Topics

- Amazon S3 Multi-Region Access Points routing states
- Using Amazon S3 Multi-Region Access Point failover controls
- Amazon S3 Multi-Region Access Point failover controls errors

Amazon S3 Multi-Region Access Points routing states

Your Amazon S3 Multi-Region Access Points failover configuration determines the routing status of the AWS Regions that are used with the Multi-Region Access Point. You can configure your Amazon S3 Multi-Region Access Point to be in an active-active state or active-passive state.

- **Active-active** – In an active-active configuration, all requests are automatically sent to the closest proximity AWS Region in your Multi-Region Access Point. After the Multi-Region Access

Point has been configured to be in an active-active state, all Regions can receive traffic. If traffic disruption occurs in an active-active configuration, network traffic will automatically be redirected to one of the active Regions.

- **Active-passive** – In an active-passive configuration, the active Regions in your Multi-Region Access Point receive traffic and the passive ones do not. If you intend to use S3 failover controls to initiate failover in a disaster situation, set up your Multi-Region Access Points in an active-passive configuration while you're testing and performing disaster-recovery planning.

Using Amazon S3 Multi-Region Access Point failover controls

This section explains how to manage and use your Amazon S3 Multi-Region Access Points failover controls by using the AWS Management Console.

There are two failover controls in the **Failover configuration** section on your Multi-Region Access Point details page in the AWS Management Console: **Edit routing status** and **Failover**. You can use these controls as follows:

- **Edit routing status** – You can manually edit the routing statuses of up to 17 AWS Regions in a single request for your Multi-Region Access Point by choosing **Edit routing status**. You can use **Edit routing status** for the following purposes:
  - To set or edit the routing statuses of one or more Regions in your Multi-Region Access Point
  - To create a failover configuration for your Multi-Region Access Point by configuring two Regions to be in an active-passive state
  - To manually fail over your Regions
  - To manually switch traffic between Regions
- **Failover** – When you initiate failover by choosing **Failover**, you are only updating the routing statuses of two Regions that are already configured to be in an active-passive state. During a failover that you initiated by choosing **Failover**, the routing statuses between the two Regions are automatically switched.

Editing the routing status of the Regions in your Multi-Region Access Point

You can manually update the routing statuses of up to 17 AWS Regions in a single request for your Multi-Region Access Point by choosing **Edit routing status** in the **Failover configuration** section on your Multi-Region Access Point details page. However, when you initiate failover by choosing **Failover**, you are only updating the routing statuses of two Regions that are already configured to

be in an active-passive state. During a failover that you initiated by choosing **Failover**, the routing statuses between the two Regions are automatically switched.

You can use **Edit routing status** (as described in the following procedure) for the following purposes:

- To set or edit the routing statuses of one or more Regions in your Multi-Region Access Point
- To create a failover configuration for your Multi-Region Access Point by configuring two Regions to be in an active-passive state
- To manually fail over your Regions
- To manually switch traffic between Regions

Using the S3 console

To update the routing status of the Regions in your Multi-Region Access Point

1. Sign in to the AWS Management Console.
2. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
3. In the left navigation pane, choose **Multi-Region Access Points**.
4. Choose the Multi-Region Access Point that you want to update.
5. Choose the **Replication and failover** tab.
6. Select one or more Regions that you want to edit the routing status of.

   > **Note**
   >
   > To initiate failover, at least one AWS Region must be designated as **Active** and one Region must be designated as **Passive** in your Multi-Region Access Point.

7. Choose **Edit routing status**.
8. In the dialog box that appears, select **Active** or **Passive** for the **Routing status** for each Region.

   An active state allows traffic to be routed to the Region. A passive state stops any traffic from being directed to the Region.

If you are creating a failover configuration for your Multi-Region Access Point or initiating failover, at least one AWS Region must be designated as **Active** and one Region must be designated as **Passive** in your Multi-Region Access Point.

9. Choose **Save routing status**. It takes about 2 minutes for traffic to be redirected.

After you submit the routing status of the AWS Regions for your Multi-Region Access Point, you can verify your routing status changes. To verify these changes, go to Amazon CloudWatch at https://console.aws.amazon.com/cloudwatch/ to monitor the shift of your Amazon S3 data-request traffic (for example, GET and PUT requests) between active and passive Regions. Any existing connections will not be terminated during failover. Existing connections will continue until they reach a success or failure status.

Using the AWS CLI

> **Note**
>
> You can run Multi-Region Access Point AWS CLI routing commands against any of these five Regions:
>
> - `ap-southeast-2`
> - `ap-northeast-1`
> - `us-east-1`
> - `us-west-2`
> - `eu-west-1`

The following example command updates your current Multi-Region Access Point route configuration. To update the active or passive status of a bucket, set the `TrafficDialPercentage` value to `100` for active and to `0` for passive. In this example, *`amzn-s3-demo-bucket1`* is set to active, and *`amzn-s3-demo-bucket2`* is set to passive. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control submit-multi-region-access-point-routes
--region ap-southeast-2
--account-id 123456789012
--mrap MultiRegionAccessPoint_ARN
--route-updates Bucket=amzn-s3-demo-bucket1,TrafficDialPercentage=100
```

```
                Bucket=amzn-s3-demo-bucket2
,TrafficDialPercentage=0
```

The following example command gets your updated Multi-Region Access Point routing configuration. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 123456789012
--mrap MultiRegionAccessPoint_ARN
```

Initiating failover

When you initiate failover by choosing **Failover** in the **Failover configuration** section on your Multi-Region Access Point details page, Amazon S3 request traffic automatically gets shifted to an alternate AWS Region. The failover process is completed within 2 minutes.

You can initiate a failover across any two AWS Regions at one time (of the 17 Regions where Multi-Region Access Points are supported). Failover events are then logged in AWS CloudTrail. Upon failover completion, you can monitor Amazon S3 traffic and any traffic routing updates to the new active Region in Amazon CloudWatch.

> **Important**
>
> To keep all metadata and objects in sync across buckets during data replication, we recommend that you create two-way replication rules and enable replica modification sync before configuring your failover controls.
> Two-way replication rules help ensure that when data is written to the Amazon S3 bucket that traffic fails over to, that data is then replicated back to the source bucket. Replica modification sync helps ensure that object metadata is also synchronized between buckets during two-way replication.
> For more information about configuring replication to support failover, see the section called “Bucket replication”.

To initiate failover between replicated buckets

1. Sign in to the AWS Management Console.

2. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
3. In the left navigation pane, choose **Multi-Region Access Points**.
4. Choose the Multi-Region Access Point that you want to use to initiate failover.
5. Choose the **Replication and failover** tab.
6. Scroll down to the **Failover configuration** section and select two AWS Regions.

> **Note**
>
> To initiate failover, at least one AWS Region must be designated as **Active** and one Region must be designated as **Passive** in your Multi-Region Access Point. An active state allows traffic to be directed to a Region. A passive state stops any traffic from being directed to the Region.

7. Choose **Failover**.
8. In the dialog box, choose **Failover** again to initiate the failover process. During this process, the routing statuses of the two Regions are automatically switched. All new traffic is directed to the Region that becomes active, and traffic stops being directed to the Region that becomes passive. It takes about 2 minutes for traffic to be redirected.

   After you initiate the failover process, you can verify your traffic changes. To verify these changes, go to Amazon CloudWatch at https://console.aws.amazon.com/cloudwatch/ to monitor the shift of your Amazon S3 data-request traffic (for example, GET and PUT requests) between active and passive Regions. Any existing connections will not be terminated during failover. Existing connections will continue until they reach a success or failure status.

Viewing your Amazon S3 Multi-Region Access Point routing controls

Using the S3 console

To view the routing controls for your Amazon S3 Multi-Region Access Point

1. Sign in to the AWS Management Console.
2. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
3. In the left navigation pane, choose **Multi-Region Access Points**.
4. Choose the Multi-Region Access Point that you want to review.

5. Choose the **Replication and failover** tab. This page displays the routing configuration details and summary for your Multi-Region Access Point, associated replication rules, and replication metrics. You can see the routing status of your Regions in the **Failover configuration** section.

Using the AWS CLI

The following example AWS CLI command gets your current Multi-Region Access Point route configuration for the specified Region. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 123456789012
--mrap MultiRegionAccessPoint_ARN
```

> **Note**
>
> This command can only be executed against these five Regions:
>
> - ap-southeast-2
> - ap-northeast-1
> - us-east-1
> - us-west-2
> - eu-west-1

Amazon S3 Multi-Region Access Point failover controls errors

When you update the failover configuration for your Multi-Region Access Point, you might encounter one of these errors:

- HTTP 400 Bad Request: This error can occur if you enter an invalid Multi-Region Access Point ARN while updating your failover configuration. You can confirm your Multi-Region Access Point ARN by reviewing your Multi-Region Access Point policy. To review or update your Multi-Region Access Point policy, see Editing the Multi-Region Access Point policy. This error can also occur if you use an empty string or a random string while updating your Amazon S3 Multi-Region Access Point failover controls. Make sure to use the Multi-Region Access Point ARN format:

`arn:aws:s3::account-id:accesspoint/MultiRegionAccessPoint_alias`

- HTTP 503 Slow Down: This error occurs if you send too many requests in a short period of time. Rejected requests will result in an error.
- HTTP 409 Conflict: This error occurs when two or more concurrent route configuration update requests are targeting a single Multi-Region Access Point. The first request succeeds, but any other requests fail with an error.
- HTTP 405 Method Not Allowed: This error occurs when you've selected a Multi-Region Access Point with only one AWS Region when initiating failover. You must select two Regions before you can initiate failover. Otherwise, an error is returned.

Configuring replication for use with Multi-Region Access Points

When you make a request to a Multi-Region Access Point endpoint, Amazon S3 automatically routes the request to the bucket that is closest to you. Amazon S3 doesn't consider the contents of the request when making this decision. If you make a request to `GET` an object, your request might be routed to a bucket that doesn't have a copy of this object. If that happens, you receive an HTTP status code 404 (Not Found) error. For more information about Multi-Region Access Point request routing, see the section called “Request routing”.

If you want the Multi-Region Access Point to be able to retrieve the object regardless of which bucket receives the request, you must configure Amazon S3 Cross-Region Replication (CRR).

For example, consider a Multi-Region Access Point with three buckets:

- A bucket named *amzn-s3-demo-bucket1* in the Region `US West (Oregon)` that contains the object `my-image.jpg`
- A bucket named *amzn-s3-demo-bucket2* in the Region `Asia Pacific (Mumbai)` that contains the object `my-image.jpg`
- A bucket named *amzn-s3-demo-bucket* in the Region `Europe (Frankfurt)` that doesn't contain the object `my-image.jpg`

In this situation, if you make a `GetObject` request for the object `my-image.jpg`, the success of that request depends upon which bucket receives your request. Because Amazon S3 doesn't consider the contents of the request, it might route your `GetObject` request to the *amzn-s3-demo-bucket* bucket if that bucket responds of closest proximity. Even though your object is in

a bucket in the Multi-Region Access Point, you will get an HTTP 404 Not Found error because the individual bucket that received your request didn't have the object.

Enabling Cross-Region Replication (CRR) helps avoid this result. With appropriate replication rules, the `my-image.jpg` object is copied over to the *`amzn-s3-demo-bucket`* bucket. Therefore, if Amazon S3 routes your request to that bucket, you can now retrieve the object.

Replication works as normal with buckets that are assigned to a Multi-Region Access Point. Amazon S3 doesn't perform any special replication handling with buckets that are in Multi-Region Access Points. For more information about configuring replication in your buckets, see Setting up live replication overview.

Recommendations for using replication with Multi-Region Access Points

For the best replication performance when working with Multi-Region Access Points, we recommend the following:

- Configure S3 Replication Time Control (S3 RTC). To replicate your data across different Regions within a predictable time frame, you can use S3 RTC. S3 RTC replicates 99.99 percent of new objects stored in Amazon S3 within 15 minutes (backed by a service-level agreement). For more information, see the section called “Using S3 Replication Time Control”. There are additional charges for S3 RTC. For information, see Amazon S3 pricing.
- Use two-way (bidirectional) replication to support keeping buckets synchronized when buckets are updated through the Multi-Region Access Point. For more information, see the section called “Create two-way replication rules for your Multi-Region Access Point”.
- Create cross-account Multi-Region Access Points to replicate data to buckets in separate AWS accounts. This approach provides account-level separation, so that data can be accessed from and replicated across different accounts in different Regions other than the source bucket. Setting up cross-account Multi-Region Access Points comes at no additional cost. If you're a bucket owner but don't own the Multi-Region Access Point, you pay only for data transfer and request costs. Multi-Region Access Point owners pay for data routing and internet-acceleration costs. For more information, see Amazon S3 pricing.
- Enable replica modification sync for each replication rule to also keep metadata changes to your objects in sync. For more information, see Enabling replica modification sync.
- Enable Amazon CloudWatch metrics to monitor replication events. CloudWatch metrics fees apply. For more information, see Amazon CloudWatch pricing.

Topics

- Create one-way replication rules for your Multi-Region Access Point
- Create two-way replication rules for your Multi-Region Access Point
- View the replication rules for your Multi-Region Access Point

Create one-way replication rules for your Multi-Region Access Point

Replication rules enable automatic and asynchronous copying of objects across buckets. A one-way replication rule helps ensure that data is fully replicated from a source bucket in one AWS Region to a destination bucket in another Region. When one-way replication is set up, a replication rule from the source bucket (*amzn-s3-demo-bucket*) to the destination bucket (*amzn-s3-demo-bucket*) is created. Like all replication rules, you can apply the one-way replication rule to the entire Amazon S3 bucket or to a subset of objects that are filtered by a prefix or object tags.

> **Important**
>
> We recommend using one-way replication if your users will only be consuming the objects in your destination buckets. If your users will be uploading or modifying the objects in your destination buckets, use two-way replication to keep all of your buckets in sync. We also recommend two-way replication if you plan to use your Multi-Region Access Point for failover. To set up two-way replication, see the section called "Create two-way replication rules for your Multi-Region Access Point".

To create a one-way replication rule for your Multi-Region Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Choose the name of your Multi-Region Access Point.
4. Choose the **Replication and failover** tab.
5. Scroll down to the **Replication rules** section, and then choose **Create replication rules**. Make sure that you have sufficient permissions to create the replication rule, or versioning will be disabled.

> **Note**
>
> You can create replication rules only for buckets in your own account. To create replication rules for external buckets, the bucket owners must create the replication rules for those buckets.

6. On the **Create replication rules** page, choose the **Replicate objects from one or more source buckets to one or more destination buckets** template.

> **Important**
>
> When you create replication rules by using this template, they replace any existing replication rules that are already assigned to the bucket.
> To add to or modify any existing replication rules instead of replacing them, go to each bucket's **Management** tab in the console, and then edit the rules in the **Replication rules** section. You can also add to or modify existing replication rules by using the AWS CLI, SDKs, or REST API. For more information, see Replication configuration file elements.

7. In the **Source and destination** section, under **Source buckets**, select one or more buckets that you want to replicate objects from. All buckets (source and destination) that are chosen for replication must have S3 Versioning enabled, and each bucket must reside in a different AWS Region. For more information about S3 Versioning, see Using versioning in Amazon S3 buckets.

   Under **Destination buckets**, select one or more buckets that you want to replicate objects to.

8. In the **Replication rule configuration** section, choose whether the replication rule will be **Enabled** or **Disabled** when it's created.

> **Note**
>
> You can't enter a name in the **Replication rule name** box. Replication rule names are generated based on your configuration when you create the replication rule.

9. In the **Scope** section, choose the appropriate scope for your replication.

   - To replicate the whole bucket, choose **Apply to all objects in the bucket**.

- To replicate a subset of the objects in the bucket, choose **Limit the scope of this rule using one or more filters**.

  You can filter your objects by using a prefix, object tags, or a combination of both.

  - To limit replication to all objects that have names that begin with the same string (for example `pictures`), enter a prefix in the **Prefix** box.

    If you enter a prefix that is the name of a folder, you must use a delimiter such as a `/` (forward slash) to indicate its level of hierarchy (for example, `pictures/`). For more information about prefixes, see Organizing objects using prefixes.

  - To replicate all objects that have one or more object tags, choose **Add tag** and enter the key-value pair in the boxes. To add another tag, repeat the procedure. For more information about object tags, see Tagging your objects.

10. Scroll down to the **Additional replication options** section, and select the replication options that you want to apply.

> **Note**
>
> We recommend that you apply the following options:
>
> - **Replication time control (RTC)** – To replicate your data across different Regions within a predictable time frame, you can use S3 Replication Time Control (S3 RTC). S3 RTC replicates 99.99 percent of new objects stored in Amazon S3 within 15 minutes (backed by a service-level agreement). For more information, see the section called “Using S3 Replication Time Control”.
> - **Replication metrics and notifications** – Enable Amazon CloudWatch metrics to monitor replication events.
> - **Delete marker replication** – Delete markers created by S3 delete operations will be replicated. Delete markers created by lifecycle rules are not replicated. For more information, see Replicating delete markers between buckets.
>
> There are additional charges for S3 RTC and CloudWatch replication metrics and notifications. For more information, see Amazon S3 Pricing and Amazon CloudWatch pricing.

11. If you're writing a new replication rule that replaces an existing one, select **I acknowledge that by choosing Create replication rules, these existing replication rules will be overwritten**.
12. Choose **Create replication rules** to create and save your new one-way replication rule.

Create two-way replication rules for your Multi-Region Access Point

Replication rules enable automatic and asynchronous copying of objects across buckets. A two-way replication rule (also known as a bidirectional replication rule) ensures that data is fully synchronized between two or more buckets in different AWS Regions. When two-way replication is set up, a replication rule from the source bucket (DOC-EXAMPLE-BUCKET-1) to the bucket containing the replicas (DOC-EXAMPLE-BUCKET-2) is created. Then, a second replication rule from the bucket containing the replicas (DOC-EXAMPLE-BUCKET-2) to the source bucket (DOC-EXAMPLE-BUCKET-1) is created.

Like all replication rules, you can apply the two-way replication rule to the entire Amazon S3 bucket or to a subset of objects filtered by a prefix or object tags. You can also keep metadata changes to your objects in sync by enabling replica modification sync for each replication rule. You can enable replica modification sync through the Amazon S3 console, the AWS CLI, the AWS SDKs, the Amazon S3 REST API, or AWS CloudFormation.

To monitor the replication progress of objects and object metadata in Amazon CloudWatch, enable S3 Replication metrics and notifications. For more information, see Monitoring progress with replication metrics and Amazon S3 event notifications.

To create a two-way replication rule for your Multi-Region Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Choose the name of the Multi-Region Access Point that you want to update.
4. Choose the **Replication and failover** tab.
5. Scroll down to the **Replication rules** section, and then choose **Create replication rules**.
6. On the **Create replication rules** page, choose the **Replicate objects among all specified buckets** template. The **Replicate objects among all specified buckets** template sets up two-way replication (with failover capabilities) for your buckets.

> **Important**
>
> When you create replication rules by using this template, they replace any existing replication rules that are already assigned to the bucket.
> To add to or modify any existing replication rules instead of replacing them, go to each bucket's **Management** tab in the console, and then edit the rules in the **Replication rules** section. You can also add to or modify existing replication rules by using the AWS CLI, AWS SDKs, or Amazon S3 REST API. For more information, see Replication configuration file elements.

7. In the **Buckets** section, select at least two buckets that you want to replicate objects from. All buckets chosen for replication must have S3 Versioning enabled, and each bucket must reside in a different AWS Region. For more information about S3 Versioning, see Using versioning in Amazon S3 buckets.

   > **Note**
   >
   > Make sure that you have the required read and replicate permissions to establish replication, or you will encounter errors. For more information, see Creating an IAM role.

8. In the **Replication rule configuration** section, choose whether the replication rule will be **Enabled** or **Disabled** when it's created.

   > **Note**
   >
   > You can't enter a name in the **Replication rule name** box. Replication rule names are generated based on your configuration when you create the replication rule.

9. In the **Scope** section, choose the appropriate scope for your replication.

   - To replicate the whole bucket, choose **Apply to all objects in the bucket**.
   - To replicate a subset of the objects in the bucket, choose **Limit the scope of this rule using one or more filters**.

     You can filter your objects by using a prefix, object tags, or a combination of both.

- To limit replication to all objects that have names that begin with the same string (for example `pictures`), enter a prefix in the **Prefix** box.

  If you enter a prefix that is the name of a folder, you must use a `/` (forward slash) as the last character (for example, `pictures/`).

- To replicate all objects that have one or more object tags, choose **Add tag** and enter the key-value pair in the boxes. To add another tag, repeat the procedure. For more information about object tags, see Tagging your objects.

10. Scroll down to the **Additional replication options** section, and select the replication options that you want to apply.

> **Note**
>
> We recommend that you apply the following options, especially if you intend to configure your Multi-Region Access Point to support failover:
>
> - **Replication time control (RTC)** – To replicate your data across different Regions within a predictable time frame, you can use S3 Replication Time Control (S3 RTC). S3 RTC replicates 99.99 percent of new objects stored in Amazon S3 within 15 minutes (backed by a service-level agreement). For more information, see the section called "Using S3 Replication Time Control".
> - **Replication metrics and notifications** – Enable Amazon CloudWatch metrics to monitor replication events.
> - **Delete marker replication** – Delete markers created by S3 delete operations will be replicated. Delete markers created by lifecycle rules are not replicated. For more information, see Replicating delete markers between buckets.
> - **Replica modification sync** – Enable replica modification sync for each replication rule to also keep metadata changes to your objects in sync. For more information, see Enabling replica modification sync.
>
> There are additional charges for S3 RTC and CloudWatch replication metrics and notifications. For more information, see Amazon S3 Pricing and Amazon CloudWatch pricing.

11. If you're writing a new replication rule that replaces an existing one, select **I acknowledge that by choosing Create replication rules, these existing replication rules will be overwritten**.

12. Choose **Create replication rules** to create and save your new two-way replication rules.

View the replication rules for your Multi-Region Access Point

With Multi-Region Access Points, you can either set up one-way replication rules or two-way (bidirectional) replication rules. For information about how to manage your replication rules, see Managing replication rules by using the Amazon S3 console.

To view the replication rules for your Multi-Region Access Point

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Multi-Region Access Points**.
3. Choose the name of your Multi-Region Access Point.
4. Choose the **Replication and failover** tab.
5. Scroll down to the **Replication rules** section. This section lists all of the replication rules that have been created for your Multi-Region Access Point.

> **Note**
>
> If you've added a bucket from another account to this Multi-Region Access Point, you must have the `s3:GetBucketReplication` permission from the bucket owner to view the replication rules for that bucket.

Using Multi-Region Access Points with supported API operations

Amazon S3 provides a set of operations to manage Multi-Region Access Points. Amazon S3 processes some of these operations synchronously and some asynchronously. When you invoke an asynchronous operation, Amazon S3 first synchronously authorizes the requested operation. If authorization is successful, Amazon S3 returns a token that you can use to track the progress and results of the requested operation.

> **Note**
>
> Requests that are made through the Amazon S3 console are always synchronous. The console waits until the request is completed before allowing you to submit another request.

You can view the current status and results of asynchronous operations by using the console, or you can use `DescribeMultiRegionAccessPointOperation` in the AWS CLI, AWS SDKs, or REST API. Amazon S3 provides a tracking token in the response to an asynchronous operation. You include that tracking token as an argument to `DescribeMultiRegionAccessPointOperation`. When you include the tracking token, Amazon S3 then returns the current status and results of the specified operation, including any errors or relevant resource information. Amazon S3 performs `DescribeMultiRegionAccessPointOperation` operations synchronously.

All control plane requests to create or maintain Multi-Region Access Points must be routed to the `US West (Oregon)` Region. For Multi-Region Access Point data plane requests, Regions don't need to be specified. For the Multi-Region Access Point failover control plane, the request must be routed to one of the five supported Regions. For more information about Multi-Region Access Point supported Regions, see Multi-Region Access Point restrictions and limitations.

In addition, you must grant the `s3:ListAllMyBuckets` permission to the user, role, or other AWS Identity and Access Management (IAM) entity that makes a request to manage a Multi-Region Access Point.

The following examples demonstrate how to use Multi-Region Access Points with compatible operations in Amazon S3.

Topics

- Multi-Region Access Point compatibility with AWS services and AWS SDKs
- Multi-Region Access Point compatibility with S3 operations
- View your Multi-Region Access Point routing configuration
- Update your underlying Amazon S3 bucket policy
- Update a Multi-Region Access Point route configuration
- Add an object to a bucket in your Multi-Region Access Point
- Retrieve objects from your Multi-Region Access Point

- List objects that are stored in a bucket underlying your Multi-Region Access Point
- Use a presigned URL with Multi-Region Access Points
- Use a bucket that's configured with Requester Pays with Multi-Region Access Points

Multi-Region Access Point compatibility with AWS services and AWS SDKs

To use a Multi-Region Access Point with applications that require an Amazon S3 bucket name, use the Amazon Resource Name (ARN) of the Multi-Region Access Point when making requests by using an AWS SDK. To check which AWS SDKs are compatible with Multi-Region Access Points, see Compatibility with AWS SDKs.

Multi-Region Access Point compatibility with S3 operations

You can use use the following Amazon S3 data plane API operations to perform actions on objects in buckets that are associated with your Multi-Region Access Point. The following S3 operations can accept Multi-Region Access Point ARNs:

- AbortMultipartUpload
- CompleteMultipartUpload
- CreateMultipartUpload
- DeleteObject
- DeleteObjectTagging
- GetObject
- GetObjectAcl
- GetObjectLegalHold
- GetObjectRetention
- GetObjectTagging
- HeadObject
- ListMultipartUploads
- ListObjectsV2
- ListParts
- PutObject
- PutObjectAcl
- PutObjectLegalHold

- PutObjectRetention
- PutObjectTagging
- RestoreObject
- UploadPart

> **Note**
>
> Multi-Region Access Points supports copy operations using Multi-Region Access Points only as a destination when using the Multi-Region Access Point ARN.

You can use the following Amazon S3 control plane operations to create and manage your Multi-Region Access Points:

- CreateMultiRegionAccessPoint
- DescribeMultiRegionAccessPointOperation
- GetMultiRegionAccessPoint
- GetMultiRegionAccessPointPolicy
- GetMultiRegionAccessPointPolicyStatus
- GetMultiRegionAccessPointRoutes
- ListMultiRegionAccessPoints
- PutMultiRegionAccessPointPolicy
- SubmitMultiRegionAccessPointRoutes

View your Multi-Region Access Point routing configuration

AWS CLI

The following example command retrieves your Multi-Region Access Point route configuration so that you can see the current routing statuses for your buckets. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control get-multi-region-access-point-routes
--region eu-west-1
--account-id 111122223333
```

```
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
```

SDK for Java

The following SDK for Java code retrieves your Multi-Region Access Point route configuration so that you can see the current routing statuses for your buckets. To use this example syntax, replace the *user input placeholders* with your own information.

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.US_EAST_1)
    .credentialsProvider(credentialsProvider)
    .build();

GetMultiRegionAccessPointRoutesRequest request =
 GetMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .build();

GetMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.getMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

The following SDK for JavaScript code retrieves your Multi-Region Access Point route configuration so that you can see the current routing statuses for your buckets. To use this example syntax, replace the *user input placeholders* with your own information.

```
const REGION = 'us-east-1'

const s3ControlClient = new S3ControlClient({
  region: REGION
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new GetMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
      })
    )
    console.log('Success', data)
```

```
    return data
  } catch (err) {
    console.log('Error', err)
  }
}

run()
```

SDK for Python (Boto3)

The following AWS SDK for Python (Boto3) code retrieves your Multi-Region Access Point route configuration so that you can see the current routing statuses for your buckets. To use this example syntax, replace the *user input placeholders* with your own information.

```
s3.get_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap)['Routes']
```

Update your underlying Amazon S3 bucket policy

To grant proper access, you must also update the underlying Amazon S3 bucket policy. The following examples delegate access control to the Multi-Region Access Point policy. After you delegate access control to the Multi-Region Access Point policy, the bucket policy is no longer used for access control when requests are made through the Multi-Region Access Point.

Here's an example bucket policy that delegates access control to the Multi-Region Access Point policy. To use this example bucket policy, replace the *user input placeholders* with your own information. To apply this policy through the AWS CLI `put-bucket-policy` command, as shown in the next example, save the policy in a file, for example, `policy.json`.

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::444455556666:root"
      },
```

```
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {
        "StringEquals": {
          "s3:DataAccessPointAccount": "444455556666"
        }
      }
    }
  ]
}
```

The following `put-bucket-policy` example command associates the updated S3 bucket policy with your S3 bucket:

```
aws s3api put-bucket-policy
  --bucket amzn-s3-demo-bucket
  --policy file:///tmp/policy.json
```

Update a Multi-Region Access Point route configuration

The following example command updates the Multi-Region Access Point route configuration. Multi-Region Access Point route commands can be run against the following five Regions:

- `ap-southeast-2`
- `ap-northeast-1`
- `us-east-1`
- `us-west-2`
- `eu-west-1`

In a Multi-Region Access Point routing configuration, you can set buckets to an active or passive routing status. Active buckets receive traffic, whereas passive buckets do not. You can set a bucket's

routing status by setting the `TrafficDialPercentage` value for the bucket to `100` for active or `0` for passive.

AWS CLI

The following example command updates your Multi-Region Access Point routing configuration. In this example, *amzn-s3-demo-bucket1* is set to active status and *amzn-s3-demo-bucket2* is set to passive. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control submit-multi-region-access-point-routes
--region ap-southeast-2
--account-id 111122223333
--mrap arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap
--route-updates Bucket=amzn-s3-demo-bucket1,TrafficDialPercentage=100
                Bucket=amzn-s3-demo-bucket2,TrafficDialPercentage=0
```

SDK for Java

The following SDK for Java code updates your Multi-Region Access Point routing configuration. To use this example syntax, replace the *user input placeholders* with your own information.

```
S3ControlClient s3ControlClient = S3ControlClient.builder()
    .region(Region.ap-southeast-2)
    .credentialsProvider(credentialsProvider)
    .build();

SubmitMultiRegionAccessPointRoutesRequest request =
 SubmitMultiRegionAccessPointRoutesRequest.builder()
    .accountId("111122223333")
    .mrap("arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap")
    .routeUpdates(
        MultiRegionAccessPointRoute.builder()
            .region("eu-west-1")
            .trafficDialPercentage(100)
            .build(),
        MultiRegionAccessPointRoute.builder()
            .region("ca-central-1")
            .bucket("111122223333")
            .trafficDialPercentage(0)
            .build()
```

```
    )
    .build();

SubmitMultiRegionAccessPointRoutesResponse response =
 s3ControlClient.submitMultiRegionAccessPointRoutes(request);
```

SDK for JavaScript

The following SDK for JavaScript code updates your Multi-Region Access Point route configuration. To use this example syntax, replace the *user input placeholders* with your own information.

```
const REGION = 'ap-southeast-2'

const s3ControlClient = new S3ControlClient({
  region: REGION
})

export const run = async () => {
  try {
    const data = await s3ControlClient.send(
      new SubmitMultiRegionAccessPointRoutesCommand({
        AccountId: '111122223333',
        Mrap: 'arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap',
        RouteUpdates: [
          {
            Region: 'eu-west-1',
            TrafficDialPercentage: 100,
          },
          {
            Region: 'ca-central-1',
            Bucket: 'amzn-s3-demo-bucket1',
            TrafficDialPercentage: 0,
          },
        ],
      })
    )
    console.log('Success', data)
    return data
  } catch (err) {
    console.log('Error', err)
  }
}
```

```
run()
```

SDK for Python (Boto3)

The following SDK for Python (Boto3) code updates your Multi-Region Access Point route configuration. To use this example syntax, replace the *user input placeholders* with your own information.

```
s3.submit_multi_region_access_point_routes(
        AccountId=111122223333,
        Mrap=arn:aws:s3::111122223333:accesspoint/abcdef0123456.mrap,
        RouteUpdates= [{
            'Bucket': amzn-s3-demo-bucket,
            'Region': ap-southeast-2,
            'TrafficDialPercentage': 10
        }])
```

Add an object to a bucket in your Multi-Region Access Point

To add an object to the bucket that's associated with the Multi-Region Access Point, you can use the PutObject operation. To keep all buckets in the Multi-Region Access Point in sync, enable Cross-Region Replication.

> **Note**
>
> To use this operation, you must have the `s3:PutObject` permission for the Multi-Region Access Point. For more information about Multi-Region Access Point permission requirements, see Permissions.

AWS CLI

The following example data plane request uploads *example.txt* to the specified Multi-Region Access Point. To use this example, replace the *user input placeholders* with your own information.

```
aws s3api put-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --key example.txt --
body example.txt
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();

PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
        .key("example.txt")
        .build();

s3Client.putObject(objectRequest, RequestBody.fromString("Hello S3!"));
```

SDK for JavaScript

```
const client = new S3Client({});

async function putObjectExample() {
    const command = new PutObjectCommand({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt",
        Body: "Hello S3!",
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python (Boto3)

```
import boto3

client = boto3.client('s3')
client.put_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
    Key='example.txt',
    Body='Hello S3!'
)
```

Retrieve objects from your Multi-Region Access Point

To retrieve objects from the Multi-Region Access Point, you can use the GetObject operation.

> **Note**
>
> To use this API operation, you must have the `s3:GetObject` permission for the Multi-Region Access Point. For more information about Multi-Region Access Point permissions requirements, see Permissions.

AWS CLI

The following example data plane request retrieves *example.txt* from the specified Multi-Region Access Point and downloads it as *downloaded_example.txt*. To use this example, replace the *user input placeholders* with your own information.

```
aws s3api get-object --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap --
key example.txt downloaded_example.txt
```

SDK for Java

```
S3Client s3 = S3Client
   .builder()
   .build();

GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example.txt")
    .build();

s3Client.getObject(getObjectRequest);
```

SDK for JavaScript

```
const client = new S3Client({})

async function getObjectExample() {
    const command = new GetObjectCommand({
```

```
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
        Key: "example.txt"
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python (Boto3)

```
import boto3

client = boto3.client('s3')
client.get_object(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap',
    Key='example.txt'
)
```

List objects that are stored in a bucket underlying your Multi-Region Access Point

To return a list of objects that are stored in a bucket underlying your Multi-Region Access Point, use the ListObjectsV2 operation. In the following example command, all objects for the specified Multi-Region Access Point are listed by using the ARN for the Multi-Region Access Point. In this case, the Multi-Region Access Point ARN is:

`arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap`

> **Note**
>
> To use this API operation, you must have the `s3:ListBucket` permission for the Multi-Region Access Point and the underlying bucket. For more information about Multi-Region Access Point permissions requirements, see Permissions.

AWS CLI

The following example data plane request lists the objects in the bucket that underlies the Multi-Region Access Point that's specified by the ARN. To use this example, replace the *user input placeholders* with your own information.

```
aws s3api list-objects-v2 --bucket
 arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap
```

SDK for Java

```
S3Client s3Client = S3Client.builder()
        .build();

String bucketName = "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap";

ListObjectsV2Request listObjectsRequest = ListObjectsV2Request
    .builder()
    .bucket(bucketName)
    .build();

 s3Client.listObjectsV2(listObjectsRequest);
```

SDK for JavaScript

```
const client = new S3Client({});

async function listObjectsExample() {
    const command = new ListObjectsV2Command({
        Bucket: "arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap",
    });

    try {
        const response = await client.send(command);
        console.log(response);
    } catch (err) {
        console.error(err);
    }
}
```

SDK for Python (Boto3)

```
import boto3

client = boto3.client('s3')
client.list_objects_v2(
    Bucket='arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap'
)
```

Use a presigned URL with Multi-Region Access Points

You can use a presigned URL to generate a URL that allows others to access your Amazon S3 buckets through an Amazon S3 Multi-Region Access Point. When you create a presigned URL, you associate it with a specific object action, such as an S3 upload (`PutObject`) or an S3 download (`GetObject`). You can share the presigned URL, and anyone with access to it can perform the action embedded in the URL as if they were the original signing user.

Presigned URLs have an expiration date. When the expiration time is reached, the URL will no longer work.

Before you use S3 Multi-Region Access Points with presigned URLs, check the AWS SDK compatibility with the SigV4A algorithm. Verify that your SDK version supports SigV4A as the signing implementation that is used to sign the global AWS Region requests. For more information about using presigned URLs with Amazon S3, see Sharing objects by using presigned URLs.

The following examples show how you can use Multi-Region Access Points with presigned URLs. To use these examples, replace the *`user input placeholders`* with your own information.

AWS CLI

```
aws s3 presign
 arn:aws:s3::123456789012:accesspoint/MultiRegionAccessPoint_alias/example-file.txt
```

SDK for Python (Boto3)

```
import logging
import boto3
from botocore.exceptions import ClientError
```

```
s3_client = boto3.client('s3',aws_access_key_id='xxx',aws_secret_access_key='xxx')
s3_client.generate_presigned_url(HttpMethod='PUT',ClientMethod="put_object",
 Params={'Bucket':'arn:aws:s3::123456789012:accesspoint/
abcdef0123456.mrap','Key':'example-file'})
```

SDK for Java

```
S3Presigner s3Presigner = S3Presigner.builder()
    .credentialsProvider(StsAssumeRoleCredentialsProvider.builder()
        .refreshRequest(assumeRole)
        .stsClient(stsClient)
        .build())
    .build();

GetObjectRequest getObjectRequest = GetObjectRequest.builder()
    .bucket("arn:aws:s3::123456789012:accesspoint/abcdef0123456.mrap")
    .key("example-file")
    .build();

GetObjectPresignRequest preSignedReq = GetObjectPresignRequest.builder()
    .getObjectRequest(getObjectRequest)
    .signatureDuration(Duration.ofMinutes(10))
    .build();

PresignedGetObjectRequest presignedGetObjectRequest =
 s3Presigner.presignGetObject(preSignedReq);
```

Note

To use SigV4A with temporary security credentials—for example, when using IAM roles—make sure that you request the temporary credentials from a Regional endpoint in AWS Security Token Service (AWS STS), instead of a global endpoint. If you use the global endpoint for AWS STS (`sts.amazonaws.com`), AWS STS will generate temporary credentials from a global endpoint, which isn't supported by Sig4A. As a result, you'll get an error. To resolve this issue, use any of the listed Regional endpoints for AWS STS.

Use a bucket that's configured with Requester Pays with Multi-Region Access Points

If an S3 bucket that is associated with your Multi-Region Access Points is configured to use Requester Pays, the requester will pay for the bucket request, the download, and any Multi-Region Access Points related costs. For more information, see Amazon S3 pricing.

Here's an example of a data plane request to a Multi-Region Access Point that is connected to a Requester Pays bucket.

AWS CLI

To download objects from a Multi-Region Access Point that is connected to a Requester Pays bucket, you must specify `--request-payer requester` as part of your get-object request. You must also specify the name of the file in the bucket and the location where the downloaded file should be stored.

```
aws s3api get-object --bucket MultiRegionAccessPoint_ARN --request-payer requester
 --key example-file-in-bucket.txt example-location-of-downloaded-file.txt
```

SDK for Java

To download objects from a Multi-Region Access Point that is connected to a Requester Pays bucket, you must specify the `RequestPayer.REQUESTER` as part of your `GetObject` request. You must also specify the name of the file in the bucket, as well as the location where it should be stored.

```
GetObjectResponse getObjectResponse = s3Client.getObject(GetObjectRequest.builder()
    .key("example-file.txt")
    .bucket("arn:aws:s3::
123456789012:accesspoint/abcdef0123456.mrap")
    .requestPayer(RequestPayer.REQUESTER)
    .build()
).response();
```

Monitoring and logging requests made through a Multi-Region Access Point to underlying resources

Amazon S3 logs requests made through Multi-Region Access Points and requests made to the API operations that manage them, such as `CreateMultiRegionAccessPoint` and

`GetMultiRegionAccessPointPolicy`. Requests made to Amazon S3 through a Multi-Region Access Point appear in your Amazon S3 server access logs and AWS CloudTrail logs with the Multi-Region Access Point hostname. An access point's hostname takes the form *`MRAP_alias`*`.accesspoint.s3-global.amazonaws.com`. For example, suppose that you have the following bucket and Multi-Region Access Point configuration:

- A bucket named `my-bucket-usw2` in the Region `us-west-2` that contains the object `my-image.jpg`.
- A bucket named `my-bucket-aps1` in the Region `ap-south-1` that contains the object `my-image.jpg`.
- A bucket named `my-bucket-euc1` in the Region `eu-central-1` that doesn't contain an object named `my-image.jpg`.
- A Multi-Region Access Point named `my-mrap` with the alias `mfzwi23gnjvgw.mrap` that is configured to fulfill requests from all three buckets.
- Your AWS account ID is `123456789012`.

A request made to retrieve `my-image.jpg` directly through any of the buckets appears in your logs with a hostname of *`bucket_name`*`.s3.`*`Region`*`.amazonaws.com`.

If you make the request through the Multi-Region Access Point instead, Amazon S3 first determines which of the buckets in the different Regions is closest. After Amazon S3 determines which bucket to use to fulfill the request, it sends the request to that bucket and logs the operation by using the Multi-Region Access Point hostname. In this example, if Amazon S3 relays the request to `my-bucket-aps1`, your logs will reflect a successful `GET` request for `my-image.jpg` from `my-bucket-aps1`, using a hostname of `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`.

Important

Multi-Region Access Points aren't aware of the data contents of the underlying buckets. Therefore, the bucket that gets the request might not contain the requested data. For example, if Amazon S3 determines that the `my-bucket-euc1` bucket is the closest, your logs will reflect a failed `GET` request for `my-image.jpg` from `my-bucket-euc1`, using a hostname of `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`. If the request was routed to `my-bucket-usw2` instead, your logs would indicate a successful `GET` request.

For more information about Amazon S3 server access logs, see Logging requests with server access logging. For more information about AWS CloudTrail, see What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

Monitoring and logging requests made to Multi-Region Access Point management API operations

Amazon S3 provides several API operations to manage Multi-Region Access Points, such as `CreateMultiRegionAccessPoint` and `GetMultiRegionAccessPointPolicy`. When you make requests to these API operations by using the AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API, Amazon S3 processes these requests asynchronously. Provided that you have the appropriate permissions for the request, Amazon S3 returns a token for these requests. You can use this token with `DescribeAsyncOperation` to help you to view the status of ongoing asynchronous operations. Amazon S3 processes `DescribeAsyncOperation` requests synchronously. To view the status of asynchronous requests, you can use the Amazon S3 console, AWS CLI, SDKs, or REST API.

> **Note**
>
> The console displays only the status of asynchronous requests that were made within the previous 14 days. To view the status of older requests, use the AWS CLI, SDKs, or REST API.

Asynchronous management operations can be in one of several states:

NEW

Amazon S3 has received the request and is preparing to perform the operation.

IN_PROGRESS

Amazon S3 is currently performing the operation.

SUCCESS

The operation succeeded. The response includes relevant information, such as the Multi-Region Access Point alias for a `CreateMultiRegionAccessPoint` request.

FAILED

The operation failed. The response includes an error message that indicates the reason for the request failure.

Using AWS CloudTrail with Multi-Region Access Points

You can use AWS CloudTrail to view, search, download, archive, analyze, and respond to account activity across your AWS infrastructure. With Multi-Region Access Points and CloudTrail logging, you can identify the following:

- Who or what took which action
- Which resources were acted upon
- When the event occurred
- Other details regarding the event

You can use this logging information to help you analyze and respond to activity that occurred through your Multi-Region Access Points.

How to set up AWS CloudTrail for Multi-Region Access Points

To enable CloudTrail logging for any operations related to creating or maintaining Multi-Region Access Points, you must configure CloudTrail logging to record the events in the US West (Oregon) Region. You must set up your logging configuration this way regardless of which Region you are in when making the request, or which Regions the Multi-Region Access Point supports. All requests to create or maintain a Multi-Region Access Point are routed through the US West (Oregon) Region. We recommend that you either add this Region to an existing trail or create a new trail that contains this Region and all the Regions associated with the Multi-Region Access Point.

Amazon S3 logs all requests made through a Multi-Region Access Point and requests made to the API operations that manage access points, such as `CreateMultiRegionAccessPoint` and `GetMultiRegionAccessPointPolicy`. When you log these requests through a Multi-Region Access Point, they appear in your AWS CloudTrail logs with the hostname of the Multi-Region Access Point. For example, if you make requests to a bucket through a Multi-Region Access Point with the alias `mfzwi23gnjvgw.mrap`, the entries in the CloudTrail log will have a hostname of `mfzwi23gnjvgw.mrap.accesspoint.s3-global.amazonaws.com`.

Multi-Region Access Points route requests to the closest bucketBecause of this behavior, when you are looking at the CloudTrail logs for a Multi-Region Access Point, you will see requests being made to the underlying buckets. Some of those requests might be direct requests to the bucket that are not routed through the Multi-Region Access Point. Keep this fact in mind when reviewing traffic. When a bucket is in a Multi-Region Access Point, requests can still be made to that bucket directly without going through the Multi-Region Access Point.

There are asynchronous events involved with creating and managing Multi-Region Access Points. Asynchronous requests don't have completion events in the CloudTrail log. For more information about asynchronous requests, see Monitoring and logging requests made to Multi-Region Access Point management API operations.

For more information about AWS CloudTrail, see What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

Retaining multiple versions of objects with S3 Versioning

Versioning in Amazon S3 is a means of keeping multiple variants of an object in the same bucket. You can use the S3 Versioning feature to preserve, retrieve, and restore every version of every object stored in your buckets. With versioning you can recover more easily from both unintended user actions and application failures. After versioning is enabled for a bucket, if Amazon S3 receives multiple write requests for the same object simultaneously, it stores all of those objects.

Versioning-enabled buckets can help you recover objects from accidental deletion or overwrite. For example, if you delete an object, Amazon S3 inserts a delete marker instead of removing the object permanently. The delete marker becomes the current object version. If you overwrite an object, it results in a new object version in the bucket. You can always restore the previous version. For more information, see Deleting object versions from a versioning-enabled bucket.

By default, S3 Versioning is disabled on buckets, and you must explicitly enable it. For more information, see Enabling versioning on buckets.

Note

- The SOAP API does not support S3 Versioning. SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features are not supported for SOAP.
- Normal Amazon S3 rates apply for every version of an object stored and transferred. Each version of an object is the entire object; it is not just a diff from the previous version. Thus, if you have three versions of an object stored, you are charged for three objects.

Unversioned, versioning-enabled, and versioning-suspended buckets

Buckets can be in one of three states:

- Unversioned (the default)
- Versioning-enabled
- Versioning-suspended

You enable and suspend versioning at the bucket level. After you version-enable a bucket, it can never return to an unversioned state. But you can *suspend* versioning on that bucket.

The versioning state applies to all (never some) of the objects in that bucket. When you enable versioning in a bucket, all new objects are versioned and given a unique version ID. Objects that already existed in the bucket at the time versioning was enabled will thereafter *always* be versioned and given a unique version ID when they are modified by future requests. Note the following:

- Objects that are stored in your bucket before you set the versioning state have a version ID of `null`. When you enable versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles the objects in future requests. For more information, see Working with objects in a versioning-enabled bucket.
- The bucket owner (or any user with appropriate permissions) can suspend versioning to stop accruing object versions. When you suspend versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles objects in future requests. For more information, see Working with objects in a versioning-suspended bucket.

Using S3 Versioning with S3 Lifecycle

To customize your data retention approach and control storage costs, use object versioning with S3 Lifecycle. For more information, see Managing the lifecycle of objects. For information about creating S3 Lifecycle configurations using the AWS Management Console, AWS CLI, AWS SDKs, or the REST API, see Setting an S3 Lifecycle configuration on a bucket.

Important

If you have an object expiration lifecycle configuration in your unversioned bucket and you want to maintain the same permanent delete behavior when you enable versioning, you must add a noncurrent expiration configuration. The noncurrent expiration lifecycle configuration manages the deletes of the noncurrent object versions in the versioning-enabled bucket. (A versioning-enabled bucket maintains one current, and zero or

more noncurrent, object versions.) For more information, see Setting an S3 Lifecycle configuration on a bucket.

For information about working with S3 Versioning, see the following topics.

Topics

- How S3 Versioning works
- Enabling versioning on buckets
- Configuring MFA delete
- Working with objects in a versioning-enabled bucket
- Working with objects in a versioning-suspended bucket
- Troubleshooting versioning

How S3 Versioning works

You can use S3 Versioning to keep multiple versions of an object in one bucket so that you can restore objects that are accidentally deleted or overwritten. For example, if you apply S3 Versioning to a bucket, the following changes occur:

- If you delete an object, instead of removing the object permanently, Amazon S3 inserts a delete marker, which becomes the current object version. You can then restore the previous version. For more information, see Deleting object versions from a versioning-enabled bucket.
- If you overwrite an object, Amazon S3 adds a new object version in the bucket. The previous version remains in the bucket and becomes a noncurrent version. You can restore the previous version.

Note

Normal Amazon S3 rates apply for every version of an object that is stored and transferred. Each version of an object is the entire object; it is not a diff from the previous version. Thus, if you have three versions of an object stored, you are charged for three objects.

Each S3 bucket that you create has a *versioning* subresource associated with it. (For more information, see General purpose buckets configuration options.) By default, your bucket is *unversioned*, and the versioning subresource stores the empty versioning configuration, as follows.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

To enable versioning, you can send a request to Amazon S3 with a versioning configuration that includes an `Enabled` status.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

To suspend versioning, you set the status value to `Suspended`.

> **Note**
>
> When you enable versioning on a bucket for the first time, it might take a short amount of time for the change to be fully propagated. While this change is propagating, you may encounter intermittent `HTTP 404 NoSuchKey` errors for requests to objects created or updated after enabling versioning. We recommend that you wait for 15 minutes after enabling versioning before issuing write operations (`PUT` or `DELETE`) on objects in the bucket.

The bucket owner and all authorized AWS Identity and Access Management (IAM) users can enable versioning. The bucket owner is the AWS account that created the bucket. For more information about permissions, see Identity and Access Management for Amazon S3.

For more information about enabling and disabling S3 Versioning by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or REST API, see the section called "Enabling versioning on buckets".

Topics

- Version IDs
- Versioning workflows

Version IDs

If you enable versioning for a bucket, Amazon S3 automatically generates a unique version ID for the object that is being stored. For example, in one bucket you can have two objects with the same key (object name) but different version IDs, such as `photo.gif` (version 111111) and `photo.gif` (version 121212).
A versioning-enabled bucket that has two objects with the same key but different version IDs.

Each object has a version ID, whether or not S3 Versioning is enabled. If S3 Versioning is not enabled, Amazon S3 sets the value of the version ID to `null`. If you enable S3 Versioning, Amazon S3 assigns a version ID value for the object. This value distinguishes that object from other versions of the same key.

When you enable S3 Versioning on an existing bucket, objects that are already stored in the bucket are unchanged. Their version IDs (`null`), contents, and permissions remain the same. After you enable S3 Versioning, each object that is added to the bucket gets a version ID, which distinguishes it from other versions of the same key.

Only Amazon S3 generates version IDs, and they cannot be edited. Version IDs are Unicode, UTF-8 encoded, URL-ready, opaque strings that are no more than 1,024 bytes long. The following is an example:

```
3sL4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY+MTRCxf3vjVBH40Nr8X8gdRQBpUMLUo
```

> **Note**
>
> For simplicity, the other examples in this topic use much shorter IDs.

Versioning workflows

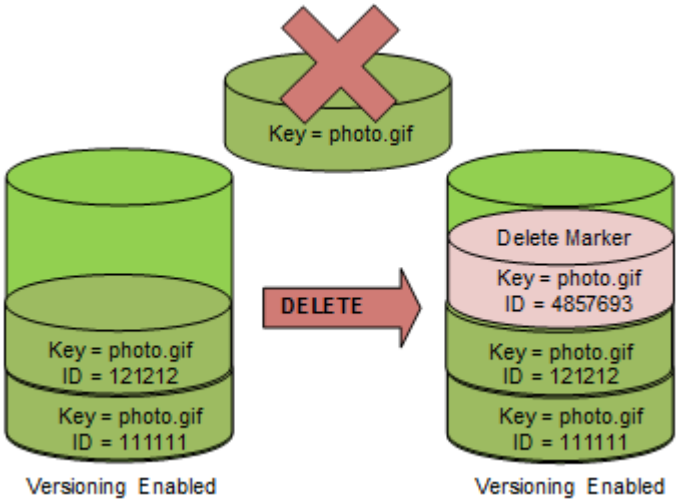
When you PUT an object in a versioning-enabled bucket, the noncurrent version is not overwritten. As shown in the following figure, when a new version of `photo.gif` is PUT into a bucket that already contains an object with the same name, the following behavior occurs:

- The original object (ID = 111111) remains in the bucket.
- Amazon S3 generates a new version ID (121212), and adds this newer version of the object to the bucket.

How S3 Versioning works when you PUT an object in a versioning-enabled bucket.

With this functionality, you can retrieve a previous version of an object if an object has been accidentally overwritten or deleted.

When you DELETE an object, all versions remain in the bucket, and Amazon S3 inserts a delete marker, as shown in the following figure.



The delete marker becomes the current version of the object. By default, GET requests retrieve the most recently stored version. Performing a `GET Object` request when the current version is a delete marker returns a `404 Not Found` error, as shown in the following figure.
A GetObject call for a delete marker returning a 404 (Not Found) error.

However, you can GET a noncurrent version of an object by specifying its version ID. In the following figure, you GET a specific object version, 111111. Amazon S3 returns that object version even though it's not the current version.

For more information, see Retrieving object versions from a versioning-enabled bucket.
How S3 Versioning works when you GET a noncurrent version in a versioning-enabled bucket.

You can permanently delete an object by specifying the version that you want to delete. Only the owner of an Amazon S3 bucket or an authorized IAM user can permanently delete a version. If your DELETE operation specifies the `versionId`, that object version is permanently deleted, and Amazon S3 doesn't insert a delete marker.
How `DELETE versionId` permanently deletes a specific object version.

You can add more security by configuring a bucket to enable multi-factor authentication (MFA) delete. When you enable MFA delete for a bucket, the bucket owner must include two forms of

authentication in any request to delete a version or change the versioning state of the bucket. For more information, see Configuring MFA delete.

When are new versions created for an object?

New versions of objects are created only when you PUT a new object. Be aware that certain actions, such as `CopyObject`, work by implementing a PUT operation.

Some actions that modify the current object don't create a new version because they don't PUT a new object. This includes actions such as changing the tags on an object.

> **Important**
>
> If you notice a significant increase in the number of HTTP 503 (Service Unavailable) responses received for Amazon S3 PUT or `DELETE` object requests to a bucket that has S3 Versioning enabled, you might have one or more objects in the bucket for which there are millions of versions. For more information, see the S3 Versioning section of Troubleshooting versioning.

Enabling versioning on buckets

You can use S3 Versioning to keep multiple versions of an object in one bucket. This section provides examples of how to enable versioning on a bucket using the console, REST API, AWS SDKs, and AWS Command Line Interface (AWS CLI).

> **Note**
>
> After enabling versioning on a bucket for the first time, it may take up to 15 minutes for the change to fully propagate across the S3 system. During this time, `GET` requests for objects created or updated after enabling versioning may result in `HTTP 404 NoSuchKey` errors. We recommend waiting 15 minutes after enabling versioning before performing any write operations (`PUT` or `DELETE`) on objects in the bucket. This waiting period helps avoid potential issues with object visibility and version tracking.

For more information about S3 Versioning, see Retaining multiple versions of objects with S3 Versioning. For information about working with objects that are in versioning-enabled buckets, see Working with objects in a versioning-enabled bucket.

To learn more about how to use S3 Versioning to protect data, see Tutorial: Protecting data on Amazon S3 against accidental deletion or application bugs using S3 Versioning, S3 Object Lock, and S3 Replication.

Each S3 bucket that you create has a *versioning* subresource associated with it. (For more information, see General purpose buckets configuration options.) By default, your bucket is *unversioned*, and the versioning subresource stores the empty versioning configuration, as follows.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</VersioningConfiguration>
```

To enable versioning, you can send a request to Amazon S3 with a versioning configuration that includes a status.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>Enabled</Status>
</VersioningConfiguration>
```

To suspend versioning, you set the status value to Suspended.

The bucket owner and all authorized users can enable versioning. The bucket owner is the AWS account that created the bucket (the root account). For more information about permissions, see Identity and Access Management for Amazon S3.

The following sections provide more detail about enabling S3 Versioning using the console, AWS CLI, and the AWS SDKs.

Using the S3 console

Follow these steps to use the AWS Management Console to enable versioning on an S3 bucket.

To enable or disable versioning on an S3 general purpose bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to enable versioning for.
4. Choose **Properties**.
5. Under **Bucket Versioning**, choose **Edit**.

6. Choose **Suspend** or **Enable**, and then choose **Save changes**.

> **Note**
>
> You can use AWS multi-factor authentication (MFA) with versioning. When you use MFA with versioning, you must provide your AWS account's access keys and a valid code from the account's MFA device to permanently delete an object version or suspend or reactivate versioning.
>
> To use MFA with versioning, you enable `MFA Delete`. However, you can't enable `MFA Delete` using the AWS Management Console. You must use the AWS Command Line Interface (AWS CLI) or the API. For more information, see Configuring MFA delete.

Using the AWS CLI

The following example enables versioning on an S3 general purpose bucket.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled
```

The following example enables S3 Versioning and multi-factor authentication (MFA) delete on a bucket for a physical MFA device. For physical MFA devices, in the `--mfa` parameter, pass a concatenation of the MFA device serial number, a space character, and the value that is displayed on your authentication device.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa "SerialNumber 123456"
```

The following example enables S3 Versioning and multi-factor authentication (MFA) delete on a bucket for a virtual MFA device. For virtual MFA devices, in the `--mfa` parameter, pass a concatenation of the MFA device ARN, a space character, and the value that is displayed on your authentication device.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa "arn:aws:iam::account-id:mfa/root-
account-mfa-device 123789"
```

> **Note**
>
> Using MFA delete requires an approved physical or virtual authentication device. For more information about using MFA delete in Amazon S3, see Configuring MFA delete.

For more information about enabling versioning using the AWS CLI, see put-bucket-versioning in the *AWS CLI Command Reference*.

Using the AWS SDKs

The following examples enable versioning on a bucket and then retrieve versioning status using the AWS SDK for Java and the AWS SDK for .NET. For information about using other AWS SDKs, see the AWS Developer Center.

.NET

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using System;
using Amazon.S3;
using Amazon.S3.Model;

namespace s3.amazon.com.docsamples
{
    class BucketVersioningConfiguration
    {
        static string bucketName = "*** bucket name ***";

        public static void Main(string[] args)
        {
            using (var client = new AmazonS3Client(Amazon.RegionEndpoint.USEast1))
            {
                try
                {
                    EnableVersioningOnBucket(client);
                    string bucketVersioningStatus =
 RetrieveBucketVersioningConfiguration(client);
                }
                catch (AmazonS3Exception amazonS3Exception)
                {
```

```
                    if (amazonS3Exception.ErrorCode != null &&
                        (amazonS3Exception.ErrorCode.Equals("InvalidAccessKeyId")
                        ||
                        amazonS3Exception.ErrorCode.Equals("InvalidSecurity")))
                    {
                        Console.WriteLine("Check the provided AWS Credentials.");
                        Console.WriteLine(
                        "To sign up for service, go to http://aws.amazon.com/s3");
                    }
                    else
                    {
                        Console.WriteLine(
                         "Error occurred. Message:'{0}' when listing objects",
                         amazonS3Exception.Message);
                    }
                }
            }

            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }

        static void EnableVersioningOnBucket(IAmazonS3 client)
        {

                PutBucketVersioningRequest request = new PutBucketVersioningRequest
                {
                    BucketName = bucketName,
                    VersioningConfig = new S3BucketVersioningConfig
                    {
                        Status = VersionStatus.Enabled
                    }
                };

                PutBucketVersioningResponse response =
client.PutBucketVersioning(request);
        }


        static string RetrieveBucketVersioningConfiguration(IAmazonS3 client)
        {
                GetBucketVersioningRequest request = new GetBucketVersioningRequest
                {
                    BucketName = bucketName
```

```
                };

                GetBucketVersioningResponse response =
 client.GetBucketVersioning(request);
                return response.VersioningConfig.Status;
            }
    }
}
```

Java

For instructions on how to create and test a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import java.io.IOException;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.AmazonS3Exception;
import com.amazonaws.services.s3.model.BucketVersioningConfiguration;
import com.amazonaws.services.s3.model.SetBucketVersioningConfigurationRequest;

public class BucketVersioningConfigurationExample {
    public static String bucketName = "*** bucket name ***";
    public static AmazonS3Client s3Client;

    public static void main(String[] args) throws IOException {
        s3Client = new AmazonS3Client(new ProfileCredentialsProvider());
        s3Client.setRegion(Region.getRegion(Regions.US_EAST_1));
        try {

            // 1. Enable versioning on the bucket.
         BucketVersioningConfiguration configuration =
           new BucketVersioningConfiguration().withStatus("Enabled");

   SetBucketVersioningConfigurationRequest setBucketVersioningConfigurationRequest
 =
     new SetBucketVersioningConfigurationRequest(bucketName,configuration);


 s3Client.setBucketVersioningConfiguration(setBucketVersioningConfigurationRequest);
```

```
   // 2. Get bucket versioning configuration information.
   BucketVersioningConfiguration conf =
 s3Client.getBucketVersioningConfiguration(bucketName);
    System.out.println("bucket versioning configuration status:    " +
 conf.getStatus());

        } catch (AmazonS3Exception amazonS3Exception) {
            System.out.format("An Amazon S3 error occurred. Exception: %s",
 amazonS3Exception.toString());
        } catch (Exception ex) {
            System.out.format("Exception: %s", ex.toString());
        }
    }
}
```

Python

The following Python code example creates an Amazon S3 bucket, enables it for versioning, and configures a lifecycle that expires noncurrent object versions after 7 days.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
 lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                   configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
```

```
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
```

```
            error,
        )

    return bucket
```

Configuring MFA delete

When working with S3 Versioning in Amazon S3 buckets, you can optionally add another layer of security by configuring a bucket to enable *MFA (multi-factor authentication) delete*. When you do this, the bucket owner must include two forms of authentication in any request to delete a version or change the versioning state of the bucket.

MFA delete requires additional authentication for either of the following operations:

- Changing the versioning state of your bucket
- Permanently deleting an object version

MFA delete requires two forms of authentication together:

- Your security credentials
- The concatenation of a valid serial number, a space, and the six-digit code displayed on an approved authentication device

MFA delete thus provides added security if, for example, your security credentials are compromised. MFA delete can help prevent accidental bucket deletions by requiring the user who initiates the delete action to prove physical possession of an MFA device with an MFA code and adding an extra layer of friction and security to the delete action.

To identify buckets that have MFA delete enabled, you can use Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Assessing your storage activity and usage with S3 Storage Lens. For a complete list of metrics, see S3 Storage Lens metrics glossary.

The bucket owner, the AWS account that created the bucket (root account), and all authorized users can enable versioning. However, only the bucket owner (root account) can enable MFA delete. For more information, see Securing Access to AWS Using MFA on the AWS Security Blog.

> **Note**
>
> To use MFA delete with versioning, you enable `MFA Delete`. However, you cannot enable `MFA Delete` using the AWS Management Console. You must use the AWS Command Line Interface (AWS CLI) or the API.
>
> For examples of using MFA delete with versioning, see the examples section in the topic Enabling versioning on buckets.
>
> You cannot use MFA delete with lifecycle configurations. For more information about lifecycle configurations and how they interact with other configurations, see How S3 Lifecycle interacts with other bucket configurations.

To enable or disable MFA delete, you use the same API that you use to configure versioning on a bucket. Amazon S3 stores the MFA delete configuration in the same *versioning* subresource that stores the bucket's versioning status.

```
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>VersioningState</Status>
  <MfaDelete>MfaDeleteState</MfaDelete>
</VersioningConfiguration>
```

To use MFA delete, you can use either a hardware or virtual MFA device to generate an authentication code. The following example shows a generated authentication code displayed on a hardware device.



MFA delete and MFA-protected API access are features intended to provide protection for different scenarios. You configure MFA delete on a bucket to help ensure that the data in your bucket cannot be accidentally deleted. MFA-protected API access is used to enforce another authentication factor (MFA code) when accessing sensitive Amazon S3 resources. You can require any operations against these Amazon S3 resources to be done with temporary credentials created using MFA. For an example, see Requiring MFA.

For more information about how to purchase and activate an authentication device, see Multi-factor authentication.

To enable S3 Versioning and configure MFA delete

Using the AWS CLI

The serial number is the number that uniquely identifies the MFA device. For physical MFA devices, this is the unique serial number that's provided with the device. For virtual MFA devices, the serial number is the device ARN. To use the following commands, replace the *`user input placeholders`* with your own information.

The following example enables S3 Versioning and multi-factor authentication (MFA) delete on a bucket for a physical MFA device. For physical MFA devices, in the `--mfa` parameter, pass a concatenation of the MFA device serial number, a space character, and the value that is displayed on your authentication device.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa "SerialNumber 123456"
```

The following example enables S3 Versioning and multi-factor authentication (MFA) delete on a bucket for a virtual MFA device. For virtual MFA devices, in the `--mfa` parameter, pass a concatenation of the MFA device ARN, a space character, and the value that is displayed on your authentication device.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket1 --versioning-
configuration Status=Enabled,MFADelete=Enabled --mfa "arn:aws:iam::account-id:mfa/root-
account-mfa-device 123789"
```

For more information, see the AWS rePost article How do I turn on MFA delete for my Amazon S3 bucket?.

Using the REST API

For more information about specifying MFA delete using the Amazon S3 REST API, see PutBucketVersioning *Amazon Simple Storage Service API Reference*.

Working with objects in a versioning-enabled bucket

Objects that are stored in an Amazon S3 bucket before you set the versioning state have a version ID of `null`. When you enable versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles the objects in future requests.

Transitioning object versions

You can define lifecycle configuration rules for objects that have a well-defined lifecycle to transition object versions to the S3 Glacier Flexible Retrieval storage class at a specific time in the object's lifetime. For more information, see Managing the lifecycle of objects.

The topics in this section explain various object operations in a versioning-enabled bucket. For more information about versioning, see Retaining multiple versions of objects with S3 Versioning.

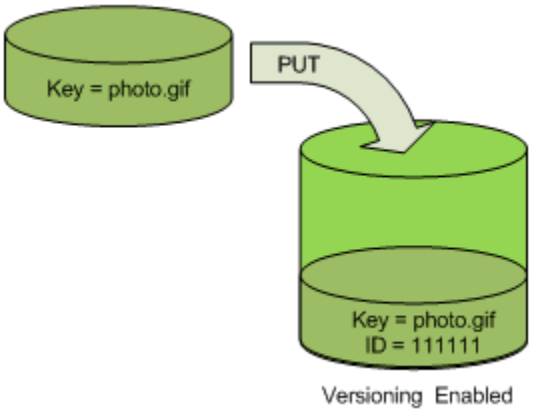
Topics

- Adding objects to versioning-enabled buckets
- Listing objects in a versioning-enabled bucket
- Retrieving object versions from a versioning-enabled bucket
- Deleting object versions from a versioning-enabled bucket
- Configuring versioned object permissions

Adding objects to versioning-enabled buckets

After you enable versioning on a bucket, Amazon S3 automatically adds a unique version ID to every object stored (using PUT, POST, or `CopyObject`) in the bucket.

The following figure shows that Amazon S3 adds a unique version ID to an object when it is added to a versioning-enabled bucket.



> **Note**
>
> The version ID values that Amazon S3 assigns are URL safe (can be included as part of a URI).

For more information about versioning, see Retaining multiple versions of objects with S3 Versioning. You can add object versions to a versioning-enabled bucket using the console, AWS SDKs, and REST API.

Using the console

For instructions, see Uploading objects.

Using the AWS SDKs

For examples of uploading objects using the AWS SDKs for Java, .NET, and PHP, see Uploading objects. The examples for uploading objects in nonversioned and versioning-enabled buckets are the same, although in the case of versioning-enabled buckets, Amazon S3 assigns a version number. Otherwise, the version number is null.

For information about using other AWS SDKs, see the AWS Developer Center.

Using the REST API

To add objects to versioning-enabled buckets

1. Enable versioning on a bucket using a `PutBucketVersioning` request.

For more information, see PutBucketVersioning in the *Amazon Simple Storage Service API Reference*.

2. Send a PUT, POST, or `CopyObject` request to store an object in the bucket.

When you add an object to a versioning-enabled bucket, Amazon S3 returns the version ID of the object in the `x-amz-version-id` response header, as shown in the following example.

```
x-amz-version-id: 3/L4kqtJlcpXroDTDmJ+rmSpXd3dIbrHY
```

Listing objects in a versioning-enabled bucket

This section provides examples of listing object versions from a versioning-enabled bucket. Amazon S3 stores object version information in the *versions* subresource that is associated with the bucket. For more information, see General purpose buckets configuration options. In order to list the objects in a versioning-enabled bucket, you need the `ListBucketVersions` permission.

Using the S3 console

Follow these steps to use the Amazon S3 console to see the different versions of an object.

To see multiple versions of an object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. To see a list of the versions of the objects in the bucket, choose the **Show versions** switch.

   For each object version, the console shows a unique version ID, the date and time the object version was created, and other properties. (Objects stored in your bucket before you set the versioning state have a version ID of **null**.)

   To list the objects without the versions, choose the **List versions** switch.

You also can view, download, and delete object versions in the object overview pane on the console. For more information, see Viewing object properties in the Amazon S3 console.

Note

To access object versions older than 300 versions, you must use the AWS CLI or the object's URL.

Important

You can undelete an object only if it was deleted as the latest (current) version. You can't undelete a previous version of an object that was deleted. For more information, see Retaining multiple versions of objects with S3 Versioning.

Using the AWS SDKs

The examples in this section show how to retrieve an object listing from a versioning-enabled bucket. Each request returns up to 1,000 versions, unless you specify a lower number. If the bucket contains more versions than this limit, you send a series of requests to retrieve the list of all versions. This process of returning results in "pages" is called *pagination*.

To show how pagination works, the examples limit each response to two object versions. After retrieving the first page of results, each example checks to determine whether the version list was truncated. If it was, the example continues retrieving pages until all versions have been retrieved.

Note

The following examples also work with a bucket that isn't versioning-enabled, or for objects that don't have individual versions. In those cases, Amazon S3 returns the object listing with a version ID of `null`.

For information about using other AWS SDKs, see the AWS Developer Center.

Java

For instructions on creating and testing a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ListVersionsRequest;
import com.amazonaws.services.s3.model.S3VersionSummary;
import com.amazonaws.services.s3.model.VersionListing;

public class ListKeysVersioningEnabledBucket {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Retrieve the list of versions. If the bucket contains more versions
            // than the specified maximum number of results, Amazon S3 returns
            // one page of results per request.
            ListVersionsRequest request = new ListVersionsRequest()
                    .withBucketName(bucketName)
                    .withMaxResults(2);
            VersionListing versionListing = s3Client.listVersions(request);
            int numVersions = 0, numPages = 0;
            while (true) {
                numPages++;
                for (S3VersionSummary objectSummary :
 versionListing.getVersionSummaries()) {
                    System.out.printf("Retrieved object %s, version %s\n",
                            objectSummary.getKey(),
                            objectSummary.getVersionId());
                    numVersions++;
                }
                // Check whether there are more pages of versions to retrieve. If
                // there are, retrieve them. Otherwise, exit the loop.
                if (versionListing.isTruncated()) {
                    versionListing =
 s3Client.listNextBatchOfVersions(versionListing);
```

```
                } else {
                    break;
                }
            }
            System.out.println(numVersions + " object versions retrieved in " +
 numPages + " pages");
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

.NET

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class ListObjectsVersioningEnabledBucketTest
    {
        static string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main(string[] args)
        {
            s3Client = new AmazonS3Client(bucketRegion);
```

```
            GetObjectListWithAllVersionsAsync().Wait();
        }

        static async Task GetObjectListWithAllVersionsAsync()
        {
            try
            {
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,
                    // You can optionally specify key name prefix in the request
                    // if you want list of object versions of a specific object.

                    // For this example we limit response to return list of 2
versions.
                    MaxKeys = 2
                };
                do
                {
                    ListVersionsResponse response = await
s3Client.ListVersionsAsync(request);
                    // Process response.
                    foreach (S3ObjectVersion entry in response.Versions)
                    {
                        Console.WriteLine("key = {0} size = {1}",
                            entry.Key, entry.Size);
                    }

                    // If response is truncated, set the marker to get the next
                    // set of keys.
                    if (response.IsTruncated)
                    {
                        request.KeyMarker = response.NextKeyMarker;
                        request.VersionIdMarker = response.NextVersionIdMarker;
                    }
                    else
                    {
                        request = null;
                    }
                } while (request != null);
            }
            catch (AmazonS3Exception e)
            {
```

```
                Console.WriteLine("Error encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

Using the REST API

Example— Listing all object versions in a bucket

To list all the versions of all the objects in a bucket, you use the `versions` subresource in a `GET Bucket` request. Amazon S3 can retrieve a maximum of 1,000 objects, and each object version counts fully as an object. Therefore, if a bucket contains two keys (for example, `photo.gif` and `picture.jpg`), and the first key has 990 versions and the second key has 400 versions, a single request would retrieve all 990 versions of `photo.gif` and only the most recent 10 versions of `picture.jpg`.

Amazon S3 returns object versions in the order in which they were stored, with the most recently stored returned first.

In a `GET Bucket` request, include the `versions` subresource.

```
GET /?versions HTTP/1.1
Host: bucketName.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 +0000
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

Example— Retrieving all versions of a key

To retrieve a subset of object versions, you use the request parameters for `GET Bucket`. For more information, see GET Bucket.

1. Set the `prefix` parameter to the key of the object that you want to retrieve.
2. Send a `GET Bucket` request using the `versions` subresource and `prefix`.

```
GET /?versions&prefix=objectName HTTP/1.1
```

Example— Retrieving objects using a prefix

The following example retrieves objects whose key is or begins with `myObject`.

```
GET /?versions&prefix=myObject HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

You can use the other request parameters to retrieve a subset of all versions of the object. For more information, see GET Bucket in the *Amazon Simple Storage Service API Reference*.

Example— Retrieving a listing of additional objects if the response is truncated

If the number of objects that could be returned in a `GET` request exceeds the value of `max-keys`, the response contains `<isTruncated>true</isTruncated>`, and includes the first key (in `NextKeyMarker`) and the first version ID (in `NextVersionIdMarker`) that satisfy the request, but were not returned. You use those returned values as the starting position in a subsequent request to retrieve the additional objects that satisfy the `GET` request.

Use the following process to retrieve additional objects that satisfy the original `GET Bucket versions` request from a bucket. For more information about `key-marker`, `version-id-marker`, `NextKeyMarker`, and `NextVersionIdMarker`, see GET Bucket in the *Amazon Simple Storage Service API Reference*.

The following are additional responses that satisfy the original `GET` request:

- Set the value of `key-marker` to the key returned in `NextKeyMarker` in the previous response.
- Set the value of `version-id-marker` to the version ID returned in `NextVersionIdMarker` in the previous response.
- Send a `GET Bucket versions` request using `key-marker` and `version-id-marker`.

Example— Retrieving objects starting with a specified key and version ID

```
GET /?versions&key-marker=myObject&version-id-marker=298459348571 HTTP/1.1
```

```
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

Using the AWS CLI

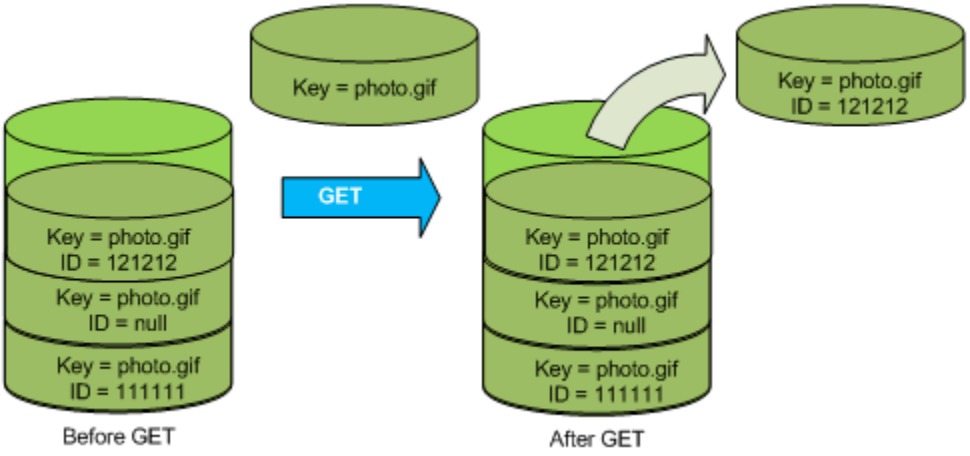
The following command returns metadata about all versions of the objects in a bucket.

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket1
```

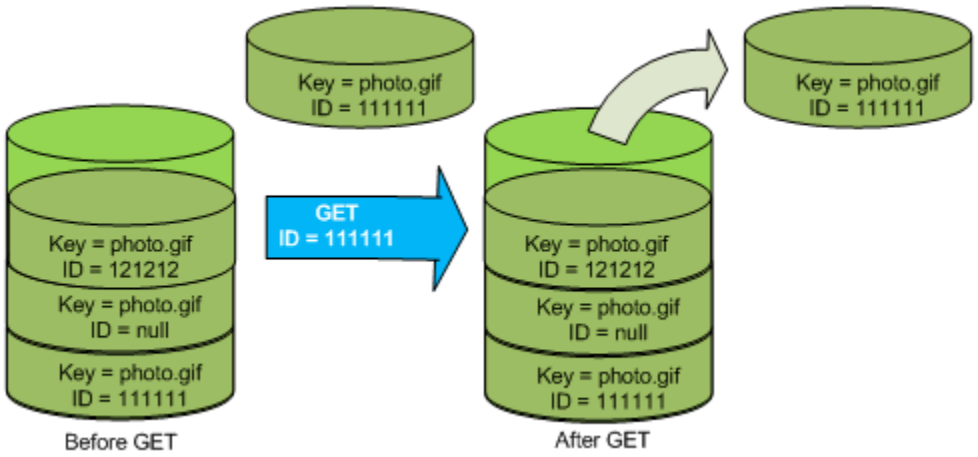
For more information about `list-object-versions` see list-object-versions in the *AWS CLI Command Reference*.

Retrieving object versions from a versioning-enabled bucket

Versioning in Amazon S3 is a way of keeping multiple variants of an object in the same bucket. A simple `GET` request retrieves the current version of an object. The following figure shows how `GET` returns the current version of the object, `photo.gif`.



To retrieve a specific version, you have to specify its version ID. The following figure shows that a `GET versionId` request retrieves the specified version of the object (not necessarily the current one).



You can retrieve object versions in Amazon S3 using the console, AWS SDKs, or REST API.

> **Note**
>
> To access object versions older than 300 versions, you must use the AWS CLI or the object's URL.

Using the S3 console

You can restore a previous version of an object using the Amazon S3 console by deleting the current version. Deleting the current version makes the previous version the new current version.

To restore a previous version by deleting the current version

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object.
4. Choose **Versions**.

   Amazon S3 shows all the versions for the object.
5. Select the check box next to the **Version ID** of the current (latest) version that you want to delete.
6. Choose **Delete**.

7. In the **Delete objects** dialog box, enter **`permanently delete`**, and then choose **Delete objects**.

   After you delete the current version, the previous version becomes the current version.

To download a previous version

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object.
4. Choose **Versions**.

   Amazon S3 shows all the versions for the object.
5. Select the check box next to the **Version ID** for the versions that you want to retrieve.
6. Choose **Actions**, choose **Download**, and save the object.

You also can view, download, and delete object versions in the object overview panel. For more information, see Viewing object properties in the Amazon S3 console.

Note

To copy a previous version in place (making it the new current version), use the AWS CLI or AWS SDKs — this approach isn't available in the Amazon S3 console. For an example, see the AWS CLI tab.

Important

You can undelete an object only if it was deleted as the latest (current) version. You can't undelete a previous version of an object that was deleted. For more information, see Retaining multiple versions of objects with S3 Versioning.

Using the AWS SDKs

The examples for uploading objects in nonversioned and versioning-enabled buckets are the same. However, for versioning-enabled buckets, Amazon S3 assigns a version number. Otherwise, the version number is null.

For examples of downloading objects using AWS SDKs for Java, .NET, and PHP, see Downloading objects.

For examples of listing the version of objects using AWS SDKs for .NET and Rust, see List the version of objects in an Amazon S3 bucket.

Using the REST API

To retrieve a specific object version

1. Set `versionId` to the ID of the version of the object that you want to retrieve.
2. Send a `GET Object versionId` request.

Example— Retrieving a versioned object

The following request retrieves version `L4kqtJlcpXroDTDmpUMLUo` of `my-image.jpg`.

```
GET /my-image.jpg?versionId=L4kqtJlcpXroDTDmpUMLUo HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

You can retrieve just the metadata of an object (not the content). For information, see the section called “Retrieving version metadata”.

For information about restoring a previous object version, see the section called “Restoring previous versions”.

Retrieving the metadata of an object version

If you only want to retrieve the metadata of an object (and not its content), you use the HEAD operation. By default, you get the metadata of the most recent version. To retrieve the metadata of a specific object version, you specify its version ID.

To retrieve the metadata of an object version

1. Set `versionId` to the ID of the version of the object whose metadata you want to retrieve.
2. Send a `HEAD Object versionId` request.

Example— Retrieving the metadata of a versioned object

The following request retrieves the metadata of version `3HL4kqCxf3vjVBH40Nrjfkd` of `my-image.jpg`.

```
HEAD /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

The following shows a sample response.

```
HTTP/1.1 200 OK
x-amz-id-2: ef8yU9AS1ed4OpIszj7UDNEHGran
x-amz-request-id: 318BC8BC143432E5
x-amz-version-id: 3HL4kqtJlcpXroDTDmjVBH40Nrjfkd
Date: Wed, 28 Oct 2009 22:32:00 GMT
Last-Modified: Sun, 1 Jan 2006 12:00:00 GMT
ETag: "fba9dede5f27731c9771645a39863328"
Content-Length: 434234
Content-Type: text/plain
Connection: close
Server: AmazonS3
```

Restoring previous versions

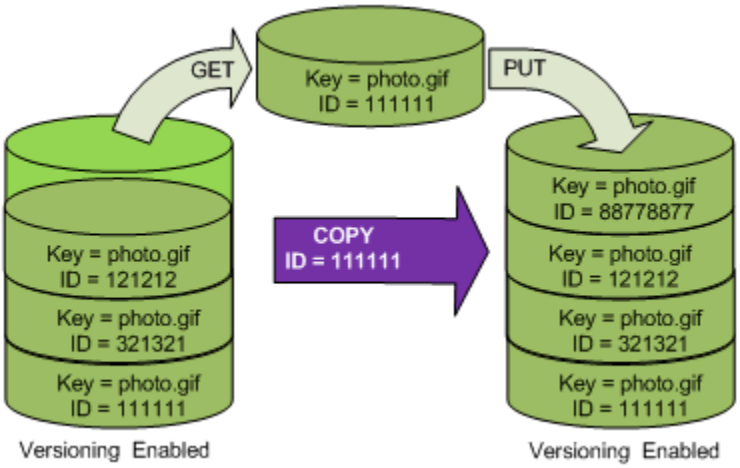
You can use versioning to retrieve previous versions of an object. There are two approaches to doing so:

- Copy a previous version of the object into the same bucket.

  The copied object becomes the current version of that object and all object versions are preserved.
- Permanently delete the current version of the object.

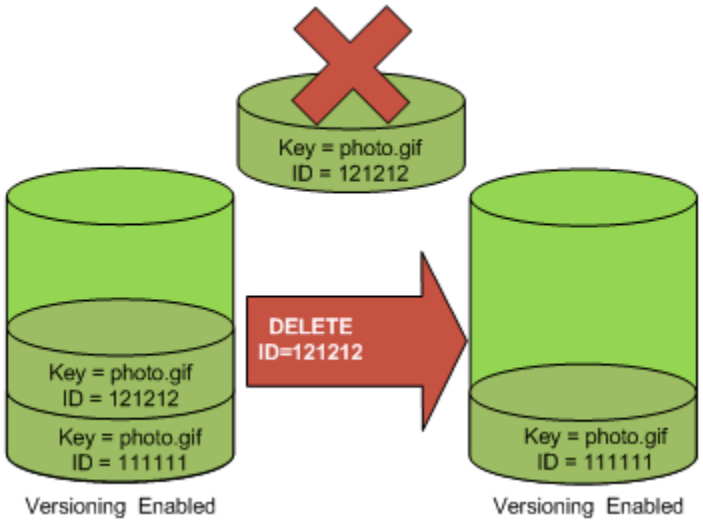
When you delete the current object version, you, in effect, turn the previous version into the current version of that object.

Because all object versions are preserved, you can make any earlier version the current version by copying a specific version of the object into the same bucket. In the following figure, the source object (ID = 111111) is copied into the same bucket. Amazon S3 supplies a new ID (88778877) and it becomes the current version of the object. So, the bucket has both the original object version (111111) and its copy (88778877). For more information about getting a previous version and then uploading it to make it the current version, see Retrieving object versions from a versioning-enabled bucket and Uploading objects.



A subsequent GET retrieves version 88778877.

The following figure shows how deleting the current version (121212) of an object leaves the previous version (111111) as the current object. For more information about deleting an object, see Deleting a single object.



A subsequent GET retrieves version 111111.

> **Note**
>
> To restore object versions in batches, you can use the `CopyObject` operation. The `CopyObject` operation copies each object that is specified in the manifest. However, be aware that objects aren't necessarily copied in the same order as they appear in the manifest. For versioned buckets, if preserving current/non-current version order is important, you should copy all non-current versions first. Then, after the first job is complete, copy the current versions in a subsequent job.

To restore previous object versions

For more guidance on restoring deleted objects, see How can I retrieve an Amazon S3 object that was deleted in a versioning-enabled bucket? in the AWS re:Post Knowledge Center.

Using the S3 console

You can restore a previous version of an object using the Amazon S3 console by deleting the current version. Deleting the current version makes the previous version the new current version.

To restore a previous version by deleting the current version

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object.
4. Choose **Versions**.

   Amazon S3 shows all the versions for the object.
5. Select the check box next to the **Version ID** of the current (latest) version that you want to delete.
6. Choose **Delete**.
7. In the **Delete objects** dialog box, enter **`permanently delete`**, and then choose **Delete objects**.

   After you delete the current version, the previous version becomes the current version.

To download a previous version

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.
3. In the **Objects** list, choose the name of the object.
4. Choose **Versions**.

   Amazon S3 shows all the versions for the object.
5. Select the check box next to the **Version ID** for the versions that you want to retrieve.
6. Choose **Actions**, choose **Download**, and save the object.

You also can view, download, and delete object versions in the object overview panel. For more information, see Viewing object properties in the Amazon S3 console.

> **Note**
>
> To copy a previous version in place (making it the new current version), use the AWS CLI or AWS SDKs — this approach isn't available in the Amazon S3 console. For an example, see the AWS CLI tab.

Important

You can undelete an object only if it was deleted as the latest (current) version. You can't undelete a previous version of an object that was deleted. For more information, see Retaining multiple versions of objects with S3 Versioning.

Using the AWS CLI

To restore a previous version by copying it in place, use the `copy-object` command. For the `--copy-source` parameter, specify the version ID of the previous version that you want to restore:

```
aws s3api copy-object \
    --bucket amzn-s3-demo-bucket \
    --key my-object.jpg \
    --copy-source amzn-s3-demo-bucket/my-object.jpg?versionId=previous-version-id
```

This creates a new current version of the object with the content from the specified previous version. Amazon S3 preserves all existing versions.

Using the AWS SDKs

For information about using other AWS SDKs, see the AWS Developer Center.

Python

The following Python code example restores a versioned object's previous version by deleting all versions that occurred after the specified rollback version.

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
```

```
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
            else:
                break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
f"{object_key}."
        )
```

Deleting object versions from a versioning-enabled bucket

You can delete object versions from Amazon S3 buckets whenever you want. You can also define lifecycle configuration rules for objects that have a well-defined lifecycle to request Amazon S3 to expire current object versions or permanently remove noncurrent object versions. When your bucket has versioning enabled or the versioning is suspended, the lifecycle configuration actions work as follows:

- The `Expiration` action applies to the current object version. Instead of deleting the current object version, Amazon S3 retains the current version as a noncurrent version by adding a *delete marker*, which then becomes the current version.
- The `NoncurrentVersionExpiration` action applies to noncurrent object versions, and Amazon S3 permanently removes these object versions. You cannot recover permanently removed objects.

For more information about S3 Lifecycle, see Managing the lifecycle of objects and Examples of S3 Lifecycle configurations.

To see how many current and noncurrent object versions that your buckets have, you can use Amazon S3 Storage Lens metrics. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. For more information, see Using S3 Storage Lens to optimize your storage costs. For a complete list of metrics, see S3 Storage Lens metrics glossary.

> **Note**
>
> Normal Amazon S3 rates apply for every version of an object that is stored and transferred, including noncurrent object versions. For more information, see Amazon S3 pricing.

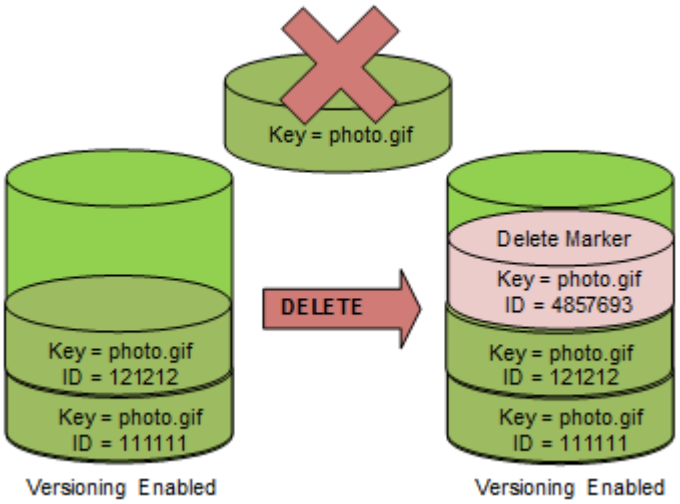
Delete request use cases

A `DELETE` request has the following use cases:

- When versioning is enabled, a simple `DELETE` cannot permanently delete an object. (A simple `DELETE` request is a request that doesn't specify a version ID.) Instead, Amazon S3 inserts a delete marker in the bucket, and that marker becomes the current version of the object with a new ID.

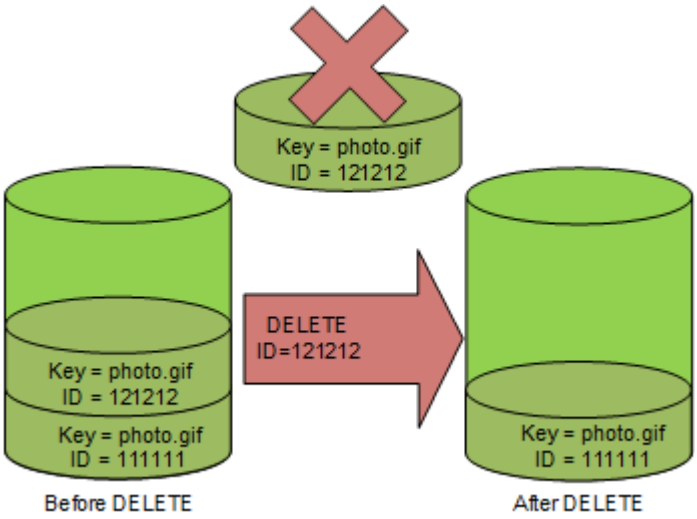
  When you try to `GET` an object whose current version is a delete marker, Amazon S3 behaves as though the object has been deleted (even though it has not been erased) and returns a 404 error. For more information, see Working with delete markers.

  The following figure shows that a simple `DELETE` does not actually remove the specified object. Instead, Amazon S3 inserts a delete marker.



- To delete versioned objects permanently, you must use `DELETE Object versionId`.

  The following figure shows that deleting a specified object version permanently removes that object.



To delete object versions

You can delete object versions in Amazon S3 using the console, AWS SDKs, the REST API, or the AWS Command Line Interface.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that contains the object.

3. In the **Objects** list, choose the name of the object.
4. Choose **Versions**.

   Amazon S3 shows all the versions for the object.
5. Select the check box next to the **Version ID** for the versions that you want to permanently delete.
6. Choose **Delete**.
7. In **Permanently delete objects?**, enter **`permanently delete`**.

   > **Warning**
   >
   > When you permanently delete an object version, the action cannot be undone.

8. Choose **Delete objects**.

   Amazon S3 deletes the object version.

Using the AWS SDKs

For examples of deleting objects using the AWS SDKs for Java, .NET, and PHP, see Deleting Amazon S3 objects. The examples for deleting objects in nonversioned and versioning-enabled buckets are the same. However, for versioning-enabled buckets, Amazon S3 assigns a version number. Otherwise, the version number is null.

For information about using other AWS SDKs, see the AWS Developer Center.

Python

The following Python code example permanently deletes a versioned object by deleting all of its versions.

```
def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
```

```
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise
```

Using the REST API

To delete a specific version of an object

- In a DELETE, specify a version ID.

Example— Deleting a specific version

The following example deletes version UIORUnfnd89493jJFJ of photo.gif.

```
DELETE /photo.gif?versionId=UIORUnfnd89493jJFJ HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:xQE0diMbLRepdf3YB+FIEXAMPLE=
Content-Type: text/plain
Content-Length: 0
```

Using the AWS CLI

The following command deletes an object named test.txt from a bucket named *amzn-s3-demo-bucket1*. To remove a specific version of an object, you must be the bucket owner and you must use the version Id subresource.

```
aws s3api delete-object --bucket amzn-s3-demo-bucket1 --key test.txt --version-
id versionID
```

For more information about delete-object see delete-object in the *AWS CLI Command Reference*.

For more information about deleting object versions, see the following topics:

- Working with delete markers
- Removing delete markers to make an older version current
- Deleting an object from an MFA delete-enabled bucket

Working with delete markers

A *delete marker* in Amazon S3 is a placeholder (or marker) for a versioned object that was specified in a simple DELETE request. A simple DELETE request is a request that doesn't specify a version ID. Because the object is in a versioning-enabled bucket, the object is not deleted. But the delete marker makes Amazon S3 behave as if the object is deleted. You can use an Amazon S3 API DELETE call on a delete marker. To do this, you must make the DELETE request by using an AWS Identity and Access Management (IAM) user or role with the appropriate permissions.

A delete marker has a *key name* (or *key*) and version ID like any other object. However, a delete marker differs from other objects in the following ways:

- A delete marker doesn't have data associated with it.
- A delete marker isn't associated with an access control list (ACL) value.
- If you issue a GET request for a delete marker, the GET request doesn't retrieve anything because a delete marker has no data. Specifically, when your GET request doesn't specify a `versionId`, you get a 404 (Not Found) error.

Delete markers accrue a minimal charge for storage in Amazon S3. The storage size of a delete marker is equal to the size of the key name of the delete marker. A key name is a sequence of Unicode characters. The UTF-8 encoding for the key name adds 1-4 bytes of storage to your bucket for each character in the name. Delete markers are stored in the S3 Standard storage class.

If you want to find out how many delete markers you have and what storage class they're stored in, you can use Amazon S3 Storage Lens. For more information, see Monitoring your storage activity and usage with Amazon S3 Storage Lens and Amazon S3 Storage Lens metrics glossary.

For more information about key names, see Naming Amazon S3 objects. For information about deleting a delete marker, see Managing delete markers.

Only Amazon S3 can create a delete marker, and it does so whenever you send a `DeleteObject` request on an object in a versioning-enabled or suspended bucket. The object specified in the DELETE request is not actually deleted. Instead, the delete marker becomes the current version of the object. The object's key name (or key) becomes the key of the delete marker.

When you get an object without specifying a `versionId` in your request, if its current version is a delete marker, Amazon S3 responds with the following:

- A 404 (Not Found) error
- A response header, `x-amz-delete-marker: true`

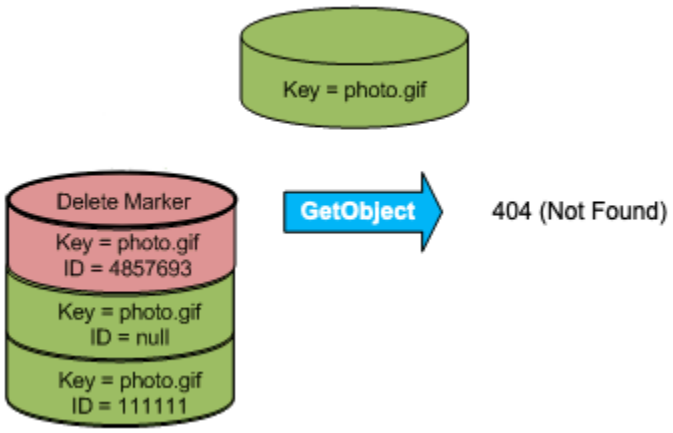
When you get an object by specifying a `versionId` in your request, if the specified version is a delete marker, Amazon S3 responds with the following:

- A 405 (Method Not Allowed) error
- A response header, `x-amz-delete-marker: true`
- A response header, `Last-Modified: timestamp` (only when using the HeadObject or GetObject API operations)

The `x-amz-delete-marker: true` response header tells you that the object accessed was a delete marker. This response header never returns `false`, because when the value is `false`, the current or specified version of the object is not a delete marker.

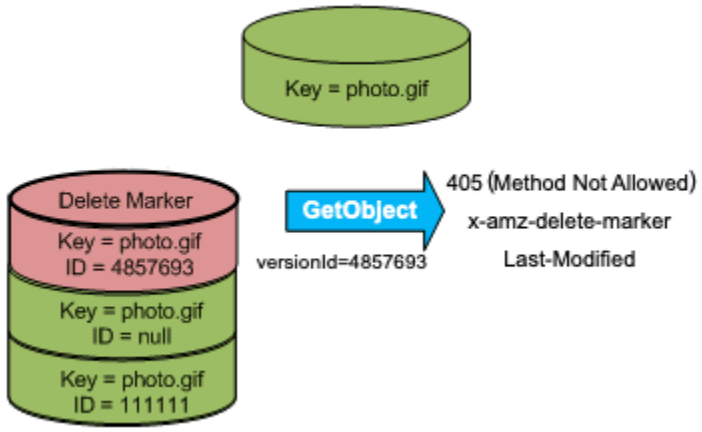
The `Last-Modified` response header provides the creation time of the delete markers.

The following figure shows how a `GetObject` API call on an object whose current version is a delete marker responds with a 404 (Not Found) error and the response header includes `x-amz-delete-marker: true`.



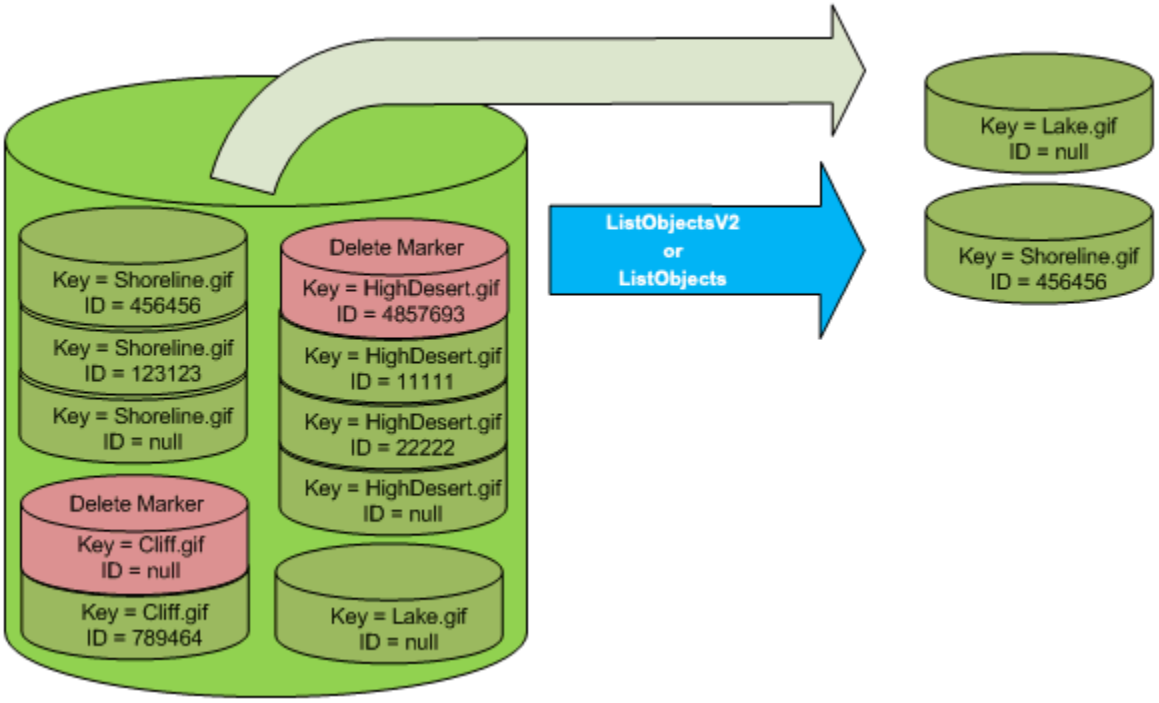
If you make a `GetObject` call on an object by specifying a `versionId` in your request, and if the specified version is a delete marker, Amazon S3 responds with a 405 (Method Not Allowed)

error and the response headers include `x-amz-delete-marker: true` and `Last-Modified: timestamp`.



Even if overwritten, delete markers remain in your object versions. The only way to list delete markers (and other versions of an object) is by using a ListObjectVersions request. You can make this request in the AWS Management Console by listing your objects in an general purpose bucket and selecting **Show versions**. For more information, see Listing objects in a versioning-enabled bucket.

The following figure shows that a ListObjectsV2 or ListObjects request doesn't return objects whose current version is a delete marker.



Managing delete markers

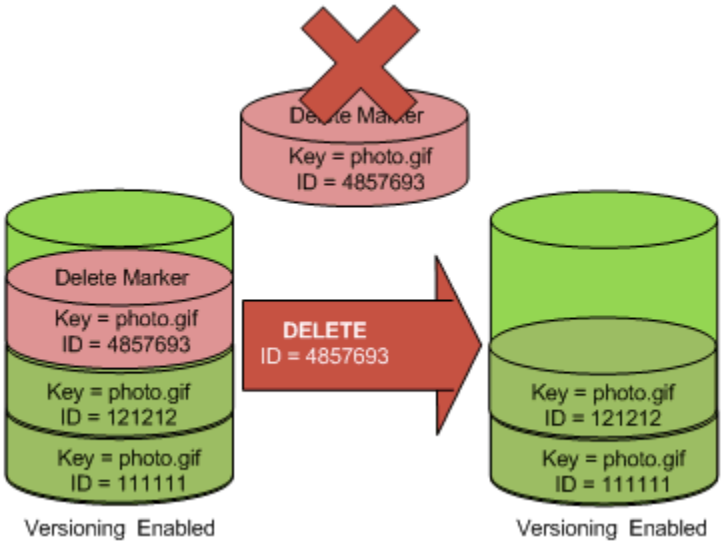
Configuring lifecycle to clean up expired delete markers automatically

An expired object delete marker is one where all object versions are deleted and only a single delete marker remains. If the lifecycle configuration is set to delete current versions, or the `ExpiredObjectDeleteMarker` action is explicitly set, Amazon S3 removes the expired object's delete marker. For an example, see Removing expired object delete markers in a versioning-enabled bucket.

Removing delete markers to make an older version current

When you delete an object in a versioning-enabled bucket, all versions remain in the bucket, and Amazon S3 creates a delete marker for the object. To undelete the object, you must delete this delete marker. For more information about versioning and delete markers, see Retaining multiple versions of objects with S3 Versioning.

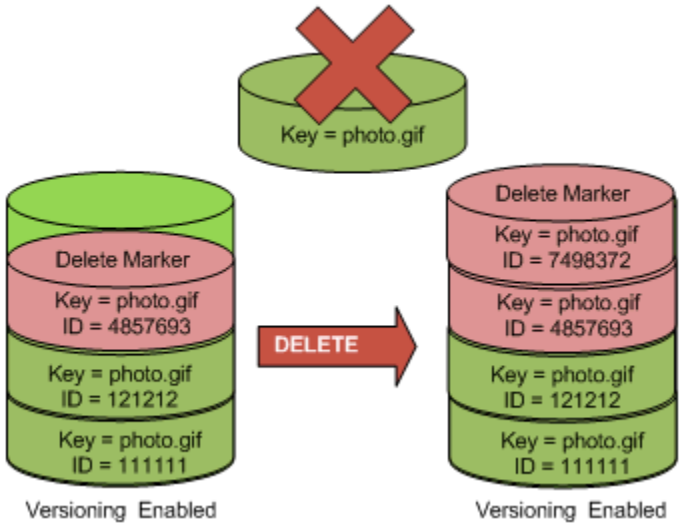
To delete a delete marker permanently, you must include its version ID in a `DeleteObject versionId` request. The following figure shows how a `DeleteObject versionId` request permanently removes a delete marker.



The effect of removing the delete marker is that a simple `GET` request will now retrieve the current version ID (121212) of the object.

> **Note**
>
> If you use a `DeleteObject` request where the current version is a delete marker (without specifying the version ID of the delete marker), Amazon S3 does not delete the delete marker, but instead PUTs another delete marker.

To delete a delete marker with a `NULL` version ID, you must pass the `NULL` as the version ID in the `DeleteObject` request. The following figure shows how a simple `DeleteObject` request made without a version ID where the current version is a delete marker, removes nothing, but instead adds an additional delete marker with a unique version ID (7498372).



Using the S3 console

Use the following steps to recover deleted objects that are not folders from your S3 bucket, including objects that are within those folders.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want.
3. To see a list of the **versions** of the objects in the bucket, choose the **List versions** switch. You'll be able to see the delete markers for deleted objects.
4. To undelete an object, you must delete the delete marker. Select the check box next to the **delete marker** of the object to recover, and then choose **Delete**.
5. Confirm the deletion on the **Delete objects** page.

a. For **Permanently delete objects?** enter **`permanently delete`**.

b. Choose **Delete objects**.

> **Note**
>
> You can't use the Amazon S3 console to undelete folders. You must use the AWS CLI or SDK. For examples, see How can I retrieve an Amazon S3 object that was deleted in a versioning-enabled bucket? in the AWS Knowledge Center.

Using the REST API

To permanently remove a delete marker

1. Set `versionId` to the ID of the version to the delete marker you want to remove.
2. Send a `DELETE Object versionId` request.

Example— Removing a delete marker

The following example removes the delete marker for `photo.gif` version 4857693.

```
DELETE /photo.gif?versionId=4857693 HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

When you delete a delete marker, Amazon S3 includes the following in the response.

```
204 NoContent
x-amz-version-id: versionID
x-amz-delete-marker: true
```

Using the AWS SDKs

For information about using other AWS SDKs, see the AWSDeveloper Center.

Python

The following Python code example demonstrates how to remove a delete marker from an object and thus makes the most recent non-current version, the current version of the object.

```
def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
 marker.
    By removing the delete marker, we make the previous version the latest version
    and the object then presents as *not* deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
 module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )
            obj = bucket.Object(object_key)
            obj.Version(latest_version["VersionId"]).delete()
            logger.info(
                "Revived %s, active version is now %s  with body '%s'",
                object_key,
                obj.version_id,
                obj.get()["Body"].read(),
            )
        else:
            logger.warning(
                "Delete marker is not the latest version for %s!", object_key
```

```
            )
    elif "Versions" in response:
        logger.warning("Got an active version for %s, nothing to do.", object_key)
    else:
        logger.error("Couldn't get any version info for %s.", object_key)
```

Deleting an object from an MFA delete-enabled bucket

When you configure MFA delete, only the root user can permanently delete object versions or change the versioning configuration on your S3 bucket. You must use an MFA device to authenticate the root user to perform the delete action.

If a bucket's versioning configuration is MFA delete enabled, the bucket owner must include the `x-amz-mfa` request header in requests to permanently delete an object version or change the versioning state of the bucket. Requests that include `x-amz-mfa` must use HTTPS.

The header's value is the concatenation of your authentication device's serial number, a space, and the authentication code displayed on it. If you don't include this request header, the request fails.

When using the AWS CLI include the same information as the value of the `mfa` parameter.

For more information about authentication devices, see Multi-factor Authentication.

For more information about enabling MFA delete, see Configuring MFA delete.

> **Note**
>
> Deleting an object in a versioning-enabled bucket that is MFA delete enabled is not available through the AWS Management Console.

Using the AWS CLI

To delete an object in a versioning-enabled bucket that is MFA delete enabled, use the following command. When you use the following example command, replace the *`user input placeholders`* with your own information.

```
 aws s3api delete-object --bucket amzn-s3-demo-bucket --key OBJECT-KEY --version-
id "VERSION ID" --mfa "MFA_DEVICE_SERIAL_NUMBER MFA_DEVICE_CODE"
```

Using the REST API

The following example deletes `my-image.jpg` (with the specified version), which is in a bucket configured with MFA delete enabled.

For more information, see DeleteObject in the Amazon Simple Storage Service API Reference

```
DELETE /my-image.jpg?versionId=3HL4kqCxf3vjVBH40Nrjfkd HTTPS/1.1
Host: bucketName.s3.amazonaws.com
x-amz-mfa: 20899872 301749
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
```

Configuring versioned object permissions

Permissions for objects in Amazon S3 are set at the version level. Each version has its own object owner. The AWS account that creates the object version is the owner. So, you can set different permissions for different versions of the same object. To do so, you must specify the version ID of the object whose permissions you want to set in a PUT `Object versionId acl` request. For a detailed description and instructions on using ACLs, see Identity and Access Management for Amazon S3.

Example— Setting permissions for an object version

The following request sets the permission of the grantee with canonical user ID *b4bf1b36f9716f094c3079dcf5ac9982d4f2847de46204d47448bc557fb5ac2a*, to `FULL_CONTROL` on the key, `my-image.jpg`, version ID, `3HL4kqtJvjVBH40Nrjfkd`.

```
PUT /my-image.jpg?acl&versionId=3HL4kqtJvjVBH40Nrjfkd HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU=
Content-Length: 124

<AccessControlPolicy>
  <Owner>
    <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
```

```
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  </AccessControlPolicy>
```

Likewise, to get the permissions of a specific object version, you must specify its version ID in a `GET Object versionId acl` request. You need to include the version ID because, by default, `GET Object acl` returns the permissions of the current version of the object.

Example— Retrieving the permissions for a specified object version

In the following example, Amazon S3 returns the permissions for the key, `my-image.jpg`, version ID, `DVBH40Nr8X8gUMLUo`.

```
GET /my-image.jpg?versionId=DVBH40Nr8X8gUMLUo&acl HTTP/1.1
Host: bucket.s3.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:0RQf4/cRonhpaBX5sCYVf1bNRuU
```

For more information, see GetObjectAcl in the *Amazon Simple Storage Service API Reference*.

Working with objects in a versioning-suspended bucket

In Amazon S3, you can suspend versioning to stop accruing new versions of the same object in a bucket. You might do this because you only want a single version of an object in a bucket. Or, you might not want to accrue charges for multiple versions.

When you suspend versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles objects in future requests. The topics in this section explain various object operations in a versioning-suspended bucket, including adding, retrieving, and deleting objects.

For more information about S3 Versioning, see Retaining multiple versions of objects with S3 Versioning. For more information about retrieving object versions, see Retrieving object versions from a versioning-enabled bucket.

Topics

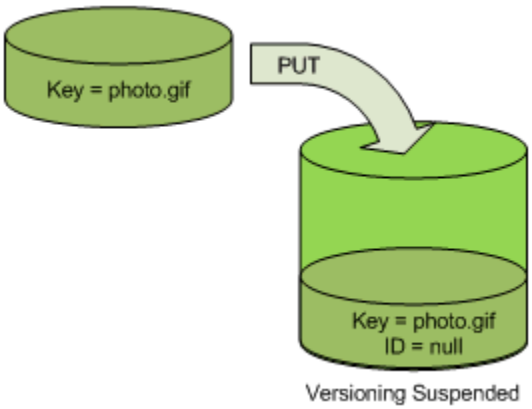
- Adding objects to versioning-suspended buckets
- Retrieving objects from versioning-suspended buckets
- Deleting objects from versioning-suspended buckets

Adding objects to versioning-suspended buckets

You can add objects to versioning-suspended buckets in Amazon S3 to create the object with a null version ID or overwrite any object version with a matching version ID.

After you suspend versioning on a bucket, Amazon S3 automatically adds a `null` version ID to every subsequent object stored thereafter (using `PUT`, `POST`, or `CopyObject`) in that bucket.

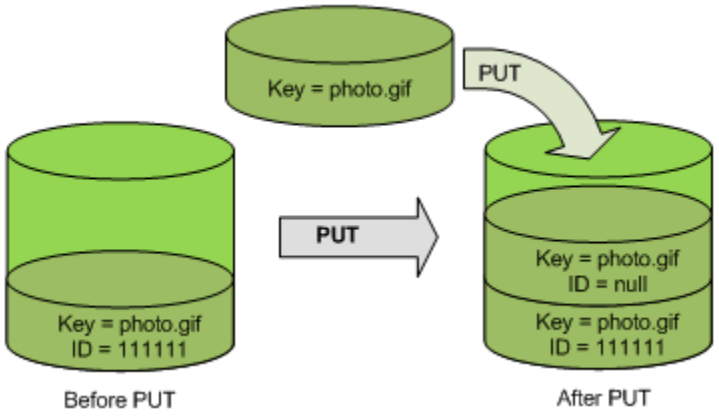
The following figure shows how Amazon S3 adds the version ID of `null` to an object when it is added to a version-suspended bucket.



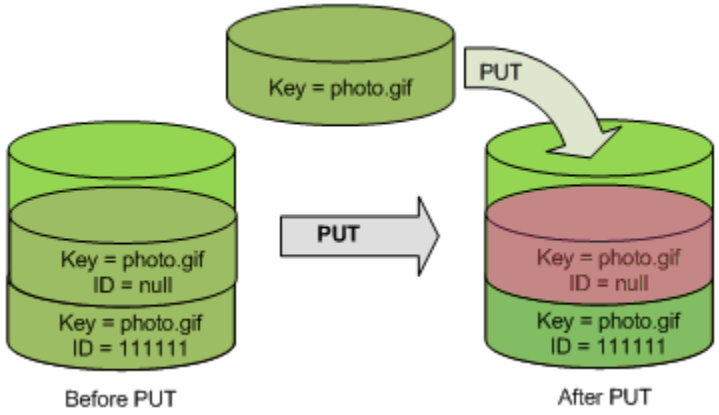
If a null version is already in the bucket and you add another object with the same key, the added object overwrites the original null version.

If there are versioned objects in the bucket, the version you PUT becomes the current version of the object. The following figure shows how adding an object to a bucket that contains versioned objects does not overwrite the object already in the bucket.

In this case, version 111111 was already in the bucket. Amazon S3 attaches a version ID of null to the object being added and stores it in the bucket. Version 111111 is not overwritten.



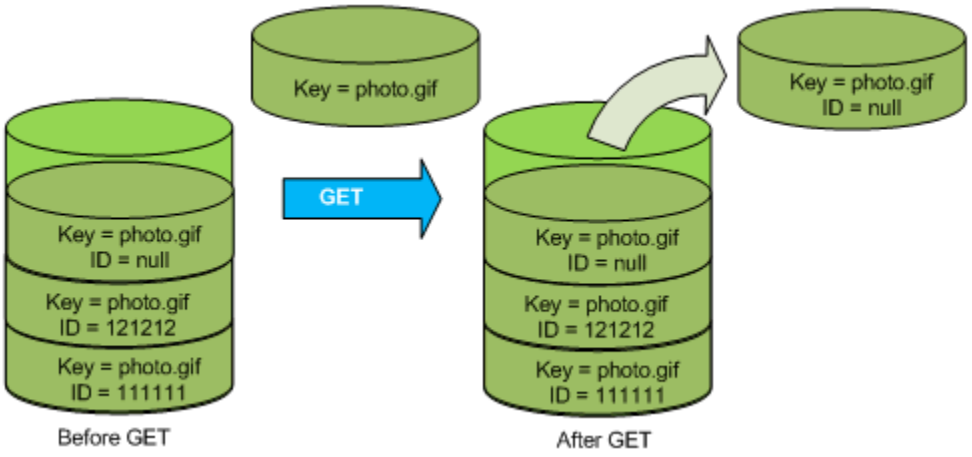
If a null version already exists in a bucket, the null version is overwritten, as shown in the following figure.



Although the key and version ID (`null`) of the null version are the same before and after the PUT, the contents of the null version originally stored in the bucket are replaced by the contents of the object PUT into the bucket.

Retrieving objects from versioning-suspended buckets

A `GET Object` request returns the current version of an object whether you've enabled versioning on a bucket or not. The following figure shows how a simple `GET` returns the current version of an object.



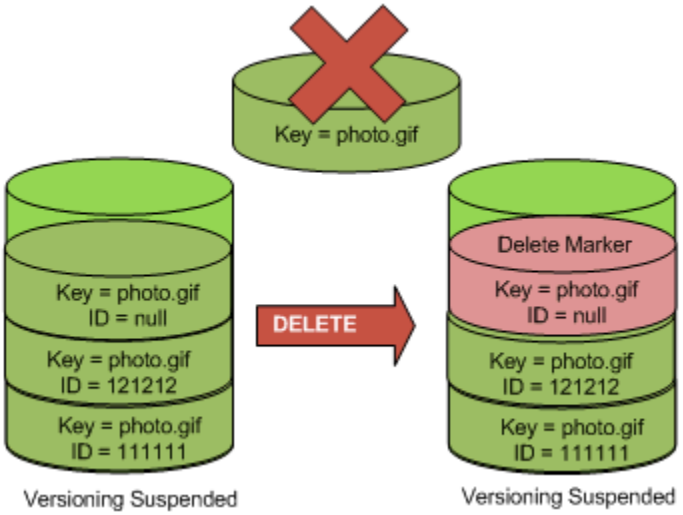
Deleting objects from versioning-suspended buckets

You can delete objects from versioning-suspended buckets to remove an object with a null version ID.

If versioning is suspended for a bucket, a `DELETE` request:

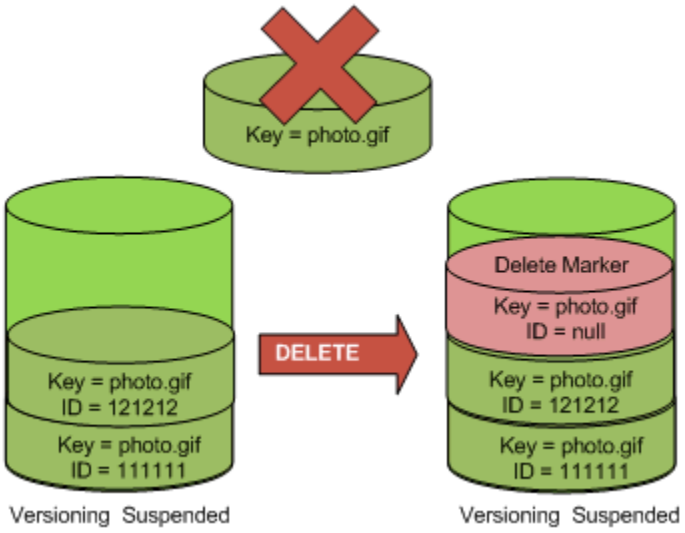
- Can only remove an object whose version ID is `null`.
- Doesn't remove anything if there isn't a null version of the object in the bucket.
- Inserts a delete marker into the bucket.

If bucket versioning is suspended, the operation removes the object that has a null `versionId`. If a version ID exists, Amazon S3 inserts a delete marker that becomes the current version of the object. The following figure shows how a simple `DELETE` removes a null version and Amazon S3 inserts a delete marker in its place instead with a `null` version ID.



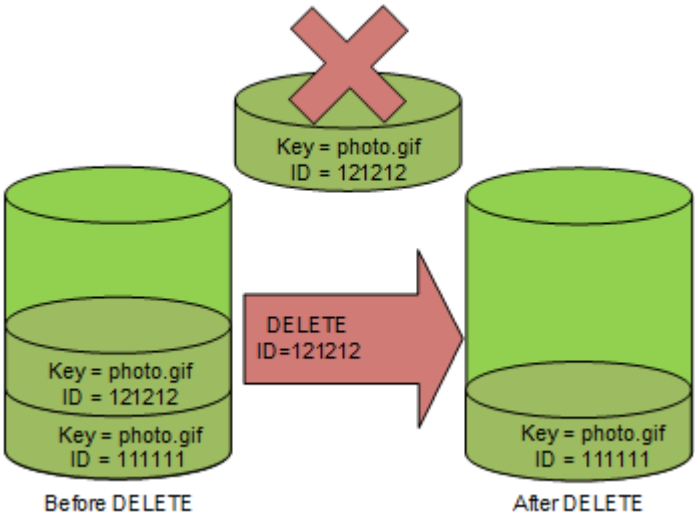
To permanently delete an object that has a `versionId`, you must include the object's `versionId` in the request. Since a delete marker doesn't contain any content, you'll lose the content for the `null` version when a delete marker replaces it.

The following figure shows a bucket that doesn't have a null version. In this case, the `DELETE` removes nothing. Instead, Amazon S3 just inserts a delete marker.



Even in a versioning-suspended bucket, the bucket owner can permanently delete a specified version by including the version ID in the `DELETE` request, unless permissions for the `DELETE` request have been explicitly denied. For example, to deny deletion of any objects that have a `null` version ID, you must explicitly deny the `s3:DeleteObject` and `s3:DeleteObjectVersions` permissions.

The following figure shows that deleting a specified object version permanently removes that version of the object. Only the bucket owner can delete a specified object version.



Troubleshooting versioning

The following topics can help you troubleshoot some common Amazon S3 versioning issues.

Topics

- I want to recover objects that were accidentally deleted in a versioning-enabled bucket
- I want to permanently delete versioned objects
- I'm experiencing performance degradation after enabling bucket versioning

I want to recover objects that were accidentally deleted in a versioning-enabled bucket

In general, when object versions are deleted from S3 buckets, there is no way for Amazon S3 to recover them. However, if you have enabled S3 Versioning on your S3 bucket, a DELETE request that doesn't specify a version ID cannot permanently delete an object. Instead, a delete marker is added as a placeholder. This delete marker becomes the current version of the object.

To verify whether your deleted objects are permanently deleted or temporarily deleted (with a delete marker in their place), do the following:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the object.

4. In the **Objects** list, Turn on the **Show versions** toggle to the right of the search bar, and then search for the deleted object in the search bar. This toggle is available only if versioning was previously enabled on the bucket.

   You can also use S3 Inventory to search for deleted objects.
5. If you can't find the object after toggling **Show versions** or creating an inventory report, and you also cannot find a delete marker of the object, the deletion is permanent and the object cannot be recovered.

You can also verify a deleted object's status by using the `HeadObject` API operation from the AWS Command Line Interface (AWS CLI). To do so, use the following `head-object` command and replace the *`user input placeholders`* with your own information:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

If you run the `head-object` command on a versioned object whose current version is a delete marker, you will receive a 404 Not Found error. For example:

An error occurred (404) when calling the HeadObject operation: Not Found

If you run the `head-object` command on a versioned object and provide the object's version ID, Amazon S3 retrieves the object's metadata, confirming that the object still exists and is not permanently deleted.

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html --
version-id versionID
```

```
{
"AcceptRanges": "bytes",
"ContentType": "text/html",
"LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
"ContentLength": 77,
"VersionId": "Zg5HyL7m.eZU9iM7AVlJkrqAiE.0UG4q",
"ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
"Metadata": {}
}
```

If the object is found and the newest version is a delete marker, the previous version of the object still exists. Because the delete marker is the current version of the object, you can recover the object by deleting the delete marker.

After you permanently remove the delete marker, the second newest version of the object becomes the current version of the object, making your object available once again. For a visual depiction of how objects are recovered, see Removing delete markers.

To remove a specific version of an object, you must be the bucket owner. To delete a delete marker permanently, you must include its version ID in a `DeleteObject` request. To delete the delete marker, use the following command, and replace the *`user input placeholders`* with your own information:

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key index.html --
version-id versionID
```

For more information about the `delete-object` command, see delete-object in the *AWS CLI Command Reference*. For more information about permanently deleting delete markers, see Managing delete markers.

I want to permanently delete versioned objects

In a versioning-enabled bucket, a `DELETE` request without a version ID cannot permanently delete an object. Instead, such a request inserts a delete marker.

To permanently delete versioned objects, you can choose from the following methods:

- Create an S3 Lifecycle rule to permanently delete noncurrent versions. To permanently delete noncurrent versions, select **Permanently delete noncurrent versions of objects**, and then enter a number under **Days after objects become noncurrent**. You can optionally specify the number of newer versions to retain by entering a value under **Number of newer versions to retain**. For more information about creating this rule, see Setting an S3 Lifecycle configuration.
- Delete a specified version by including the version ID in the `DELETE` request. For more information, see How to delete versioned objects permanently.
- Create a lifecycle rule to expire current versions. To expire current versions of objects, select **Expire current versions of objects**, and then add a number under **Days after object creation**. For more information about creating this lifecycle rule, see Setting an S3 Lifecycle configuration.
- To permanently delete all versioned objects and delete markers, create two lifecycle rules: one to expire current versions and permanently delete noncurrent versions of objects, and the other to delete expired object delete markers.

In a versioning-enabled bucket, a DELETE request that doesn't specify a version ID can remove only objects with a NULL version ID. If the object was uploaded when versioning was enabled, a DELETE request that doesn't specify a version ID creates a delete marker of that object.

> **Note**
>
> For S3 Object Lock-enabled buckets, a DELETE object request with a protected object version ID causes a 403 Access Denied error. A DELETE object request without a version ID adds a delete marker as the newest version of the object with a 200 OK response. Objects protected by Object Lock cannot be permanently deleted until their retention periods and legal holds are removed. For more information, see the section called “How S3 Object Lock works”.

I'm experiencing performance degradation after enabling bucket versioning

Performance degradation can occur on versioning-enabled buckets if there are too many delete markers or versioned objects, and if best practices aren't followed.

Too many delete markers

After you enable versioning on a bucket, a DELETE request without a version ID made to an object creates a delete marker with a unique version ID. Lifecycle configurations with an **Expire current versions of objects** rule add a delete marker with a unique version ID to every object. Excessive delete markers can reduce performance in the bucket.

When versioning is suspended on a bucket, Amazon S3 marks the version ID as NULL on newly created objects. An expiration action in a versioning-suspended bucket causes Amazon S3 to create a delete marker with NULL as the version ID. In a versioning-suspended bucket, a NULL delete marker is created for any delete request. These NULL delete markers are also called expired object delete markers when all object versions are deleted and only a single delete marker remains. If too many NULL delete markers accumulate, performance degradation in the bucket occurs.

Too many versioned objects

If a versioning-enabled bucket contains objects with millions of versions, an increase in 503 Service Unavailable errors can occur. If you notice a significant increase in the number of HTTP 503 Service Unavailable responses received for PUT or DELETE object requests to a versioning-enabled bucket, you might have one or more objects in the bucket with millions of versions. When you have objects with millions of versions, Amazon S3 automatically throttles requests to the bucket. Throttling

requests protects your bucket from an excessive amount of request traffic, which could potentially impede other requests made to the same bucket.

To determine which objects have millions of versions, use S3 Inventory. S3 Inventory generates a report that provides a flat file list of the objects in a bucket. For more information, see Cataloging and analyzing your data with S3 Inventory.

To verify if there are high number of versioned objects in the bucket, use S3 Storage Lens metrics to view the **Current version object count**, **Noncurrent version object count**, and **Delete marker object count**. For more information about Storage Lens metrics, see Amazon S3 Storage Lens metrics glossary.

The Amazon S3 team encourages customers to investigate applications that repeatedly overwrite the same object, potentially creating millions of versions for that object, to determine whether the application is working as intended. For instance, an application overwriting the same object every minute for a week can create over ten thousand versions. We recommend storing less than one hundred thousand versions for each object. If you have a use case that requires millions of versions for one or more objects, contact the AWS Support team for assistance with determining a better solution.

Best practices

To prevent versioning-related performance degradation issues, we recommend that you employ the following best practices:

- Enable a lifecycle rule to expire the previous versions of objects. For example, you can create a lifecycle rule to expire noncurrent versions after 30 days of the object being noncurrent. You can also retain multiple noncurrent versions if you don't want to delete all of them. For more information, see Setting an S3 Lifecycle configuration.
- Enable a lifecycle rule to delete expired object delete markers that don't have associated data objects in the bucket. For more information, see Removing expired object delete markers.

For additional Amazon S3 performance-optimization best practices, see Best practices design patterns.

Locking objects with Object Lock

S3 Object Lock can help prevent Amazon S3 objects from being deleted or overwritten for a fixed or variable amount of time, or indefinitely. Object Lock uses a *write-once-read-many* (WORM)

model to store objects. You can use Object Lock to help meet regulatory requirements that require WORM storage, or to add another layer of protection against object changes or deletion.

> **Note**
>
> S3 Object Lock has been assessed by Cohasset Associates for use in environments that are subject to SEC 17a-4, CFTC, and FINRA regulations. For more information about how Object Lock relates to these regulations, see the Cohasset Associates Compliance Assessment.

Object Lock provides two ways to manage object retention: *retention periods* and *legal holds*. An object version can have a retention period, a legal hold, or both.

- **Retention period** – A retention period specifies a fixed period of time during which an object version remains locked. S3 Object Lock provides two retention types: fixed and variable. With fixed retention, you specify the retain-until-date. With variable retention, you specify an event hold and an event hold duration, and Amazon S3 sets the retain-until-date when you release the hold. You can set a unique retention period for individual objects. Additionally, you can set a default retention period on an S3 bucket. You may also restrict the minimum and maximum allowable retention periods with the `s3:object-lock-remaining-retention-days` condition key in the bucket policy. This condition key helps you establish the allowable retention period. For more information, see Setting limits on retention periods with a bucket policy.
- **Legal hold** – A legal hold provides the same protection as a retention period, but it has no expiration date. Instead, a legal hold remains in place until you explicitly remove it. Legal holds are independent from retention periods and are placed on individual object versions.

Object Lock works only in buckets that have S3 Versioning enabled. When you lock an object version, Amazon S3 stores the lock information in the metadata for that object version. Placing a retention period or a legal hold on an object protects only the version that's specified in the request. Retention periods and legal holds don't prevent new versions of the object from being created, or delete markers to be added on top of the object. For information about S3 Versioning, see Retaining multiple versions of objects with S3 Versioning.

If you put an object into a bucket that already contains an existing protected object with the same object key name, Amazon S3 creates a new version of that object. The existing protected version of the object remains locked according to its retention configuration.

Note

Objects protected by Object Lock (both GOVERNANCE and COMPLIANCE modes) do not allow annotation modifications (create, update, or delete). The `BypassGovernanceRetention` permission does not apply to annotation operations. To add annotations to a locked object, create a new object version. The new version is not subject to the previous version's retention settings unless you explicitly apply retention to it.

How S3 Object Lock works

Topics

- Retention periods
- Retention modes
- Retain-until-date behavior with variable retention
- Legal holds
- How deletes work with S3 Object Lock
- Best practices for using S3 Object Lock
- Required permissions

Retention periods

A *retention period* protects an object version for a fixed or variable amount of time. Each retention period has a retain-until-date that indicates when the protection expires. After the retention period expires, the object version can be overwritten or deleted.

You can place a retention period explicitly on an individual object version or on a bucket's properties so that it applies to all objects in the bucket automatically. When you apply a fixed retention period to an object version explicitly, you specify a *Retain Until Date* for the object version, and Amazon S3 stores that date in the object version's metadata.

Object Lock provides two types of retention periods: *fixed* and *variable*. With fixed retention, you specify the retain-until-date when you apply the retention period. With *variable retention*, you defer the retain-until-date. You turn on an *event hold* and specify an *event hold duration*, and the

object stays protected while the event hold is on. When you release the hold, Amazon S3 sets the retain-until-date to the release time plus the duration.

You can also specify a retain-until-date with variable retention. Amazon S3 treats that date as a minimum, so the object stays protected until at least that date, even if you release the event hold earlier.

You can also set a retention period in a bucket's properties. When you set a retention period on a bucket, you specify a duration, in either days or years, for how long to protect every object version placed in the bucket. When you place an object in the bucket, Amazon S3 calculates a *Retain Until Date* for the object version by adding the specified duration to the object version's creation timestamp. The object version is then protected exactly as though you explicitly placed an individual lock with that retention period on the object version. A bucket's default retention can also turn on an event hold, so that every object version placed in the bucket starts with variable retention instead of a fixed retain-until-date.

> **Note**
>
> When you PUT an object version that has an explicit individual retention mode and period in a bucket, the object version's individual Object Lock settings override any bucket property retention settings.

Like all other Object Lock settings, retention periods apply to individual object versions. Different versions of a single object can have different retention modes and periods.

For example, suppose that you have an object that is 15 days into a 30-day retention period, and you PUT an object into Amazon S3 with the same name and a 60-day retention period. In this case, your PUT request succeeds, and Amazon S3 creates a new version of the object with a 60-day retention period. The older version maintains its original retention period and becomes deletable in 15 days.

After you've applied a retention setting to an object version, you can extend the retention period. To do this, submit a new Object Lock request for the object version with a *Retain Until Date* that is later than the one currently configured for the object version. Amazon S3 replaces the existing retention period with the new, longer period. Any user with permissions to place an object retention period can extend a retention period for an object version. To set a retention period, you must have the `s3:PutObjectRetention` permission.

While an event hold is on, you can also change the event hold duration. Amazon S3 does not set the retain-until-date to an earlier date than its current value.

When you set a retention period on an object or S3 bucket, you must select one of two retention modes: *compliance* or *governance*.

Retention modes

S3 Object Lock provides two retention modes that apply different levels of protection to your objects:

- Compliance mode
- Governance mode

In *compliance* mode, a protected object version can't be overwritten or deleted by any user, including the root user in your AWS account. When an object is locked in compliance mode, its retention mode can't be changed, and its retention period can't be shortened. Compliance mode helps ensure that an object version can't be overwritten or deleted for the duration of the retention period.

> **Note**
>
> The only way to delete an object under the compliance mode before its retention date expires is to delete the associated AWS account.

In *governance* mode, users can't overwrite or delete an object version or alter its lock settings unless they have special permissions. With governance mode, you protect objects against being deleted by most users, but you can still grant some users permission to alter the retention settings or delete the objects if necessary. You can also use governance mode to test retention-period settings before creating a compliance-mode retention period.

To override or remove governance-mode retention settings, you must have the `s3:BypassGovernanceRetention` permission and must explicitly include `x-amz-bypass-governance-retention:true` as a request header with any request that requires overriding governance mode.

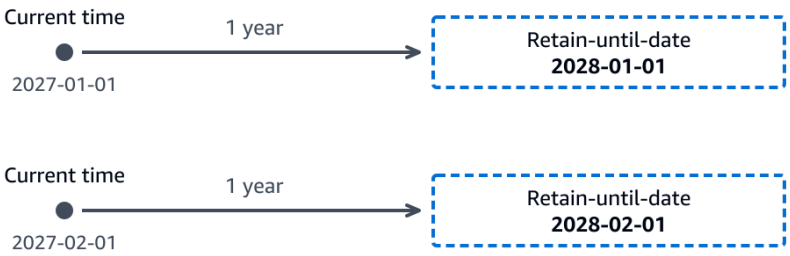
Note

By default, the Amazon S3 console includes the `x-amz-bypass-governance-retention:true` header. If you try to delete objects protected by *governance* mode and have the `s3:BypassGovernanceRetention` permission, the operation will succeed.

Retain-until-date behavior with variable retention

The retain-until-date tells you when an object can become deletable. With fixed retention, this date doesn't change. With variable retention, Amazon S3 computes the date while the event hold is on, and fixes it when you release the hold. The following examples describe this behavior in each case.

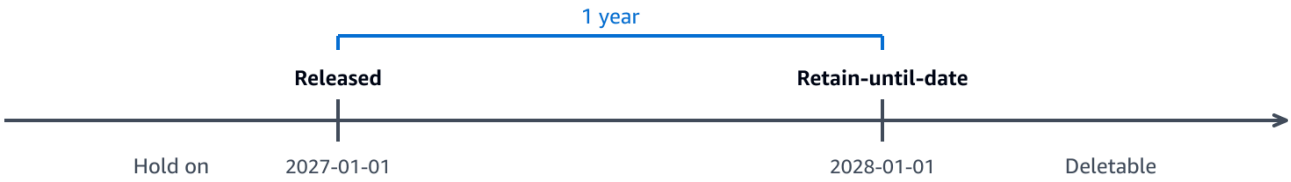
While the event hold is on, Amazon S3 computes the retain-until-date as the current time plus the duration. The date moves forward as time passes. For example, suppose that the event hold duration is 1 year. If you retrieve the retain-until-date when the current time is `2027-01-01`, the date is `2028-01-01`. If you retrieve it again when the current time is `2027-02-01`, the date is `2028-02-01`.



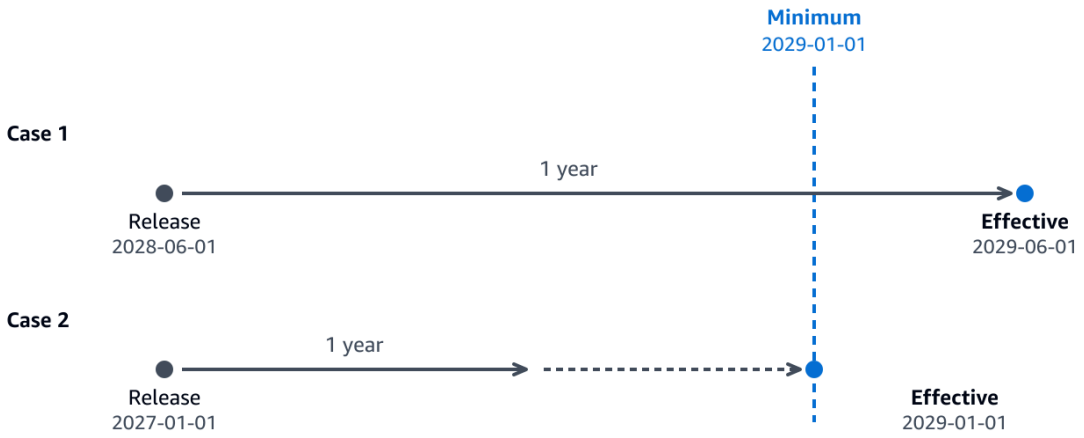
You can change the event hold duration. If you decrease it, Amazon S3 does not set the retain-until-date to an earlier date than its current value. For example, suppose that a 2-year duration set the retain-until-date to `2029-01-01`. If you decrease the duration to 1 year, the computed date is `2028-01-01`, but Amazon S3 doesn't apply it. The retain-until-date remains `2029-01-01`.



When you release the event hold, Amazon S3 fixes the retain-until-date at the release time plus the duration, and the date stops changing. For example, with a 1-year duration, releasing the event hold on `2027-01-01` fixes the retain-until-date at `2028-01-01`. Until that date, the object version remains protected. After that date, you can delete it.



If you specified a retain-until-date, the effective date is the later of that minimum date and the release time plus the duration. For example, suppose that the minimum retain-until-date is `2029-01-01` and the event hold duration is 1 year. If you release the event hold on `2028-06-01`, the computed date of `2029-06-01` is later than the minimum, so Amazon S3 applies it. If you release the event hold on `2027-01-01`, the computed date of `2028-01-01` is earlier than the minimum, so Amazon S3 applies the minimum date of `2029-01-01`.



Legal holds

With Object Lock, you can also place a *legal hold* on an object version. Like a retention period, a legal hold prevents an object version from being overwritten or deleted. However, a legal hold doesn't have an associated fixed amount of time and remains in effect until removed. Legal holds can be freely placed and removed by any user who has the `s3:PutObjectLegalHold` permission.

Legal holds are independent from retention periods. Placing a legal hold on an object version doesn't affect the retention mode or retention period for that object version.

For example, suppose that you place a legal hold on an object version and that object version is also protected by a retention period. If the retention period expires, the object doesn't lose its WORM protection. Rather, the legal hold continues to protect the object until an authorized user explicitly removes the legal hold. Similarly, if you remove a legal hold while an object version has a retention period in effect, the object version remains protected until the retention period expires.

How deletes work with S3 Object Lock

If your bucket has S3 Object Lock enabled and the object is protected by a retention period or legal hold and you try to delete an object, Amazon S3 returns one of the following responses, depending on how you tried to delete the object:

- **Permanent `DELETE` request** – If you issued a permanent `DELETE` request (a request that specifies a version ID), Amazon S3 returns an Access Denied (`403 Forbidden`) error when you try to delete the object. For more information about troubleshooting Access Denied errors with Object Lock, see the section called “S3 Object Lock settings”.

- **Simple DELETE request** – If you issued a simple DELETE request (a request that doesn't specify a version ID), Amazon S3 returns a 200 OK response and inserts a delete marker in the bucket, and that marker becomes the current version of the object with a new ID. For more information about managing delete markers with Object Lock, see the section called "Managing delete markers with Object Lock".

Best practices for using S3 Object Lock

Consider using *Governance mode* if you want to protect objects from being deleted by most users during a pre-defined retention period, but at the same time want some users with special permissions to have the flexibility to alter the retention settings or delete the objects.

Consider using *Compliance mode* if you never want any user, including the root user in your AWS account, to be able to delete the objects during a pre-defined retention period. You can use this mode in case you have a requirement to store compliant data.

You can use *Legal Hold* when you are not sure for how long you want your objects to stay immutable. This could be because you have an upcoming external audit of your data and want to keep objects immutable till the audit is complete. Alternately, you may have an ongoing project utilizing a dataset that you want to keep immutable until the project is complete.

Consider using *variable retention* when you need to protect objects but don't know the retain-until-date at write time. For example, use it when you want a configurable recovery window against ransomware or accidental deletion. You can also use it when your retention period must begin on a future business event, such as contract completion, account closure, or claim resolution.

Required permissions

Object Lock operations require specific permissions. Depending on the exact operation that you're attempting, you might need any of the following permissions:

- `s3:BypassGovernanceRetention`
- `s3:GetBucketObjectLockConfiguration`
- `s3:GetObjectLegalHold`
- `s3:GetObjectRetention`
- `s3:PutBucketObjectLockConfiguration`
- `s3:PutObjectLegalHold`
- `s3:PutObjectRetention`

For a complete list of Amazon S3 permissions with descriptions, see Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*.

For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.

For information about using conditions with permissions, see Bucket policy examples using condition keys.

Object Lock considerations

Amazon S3 Object Lock can help prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely.

You can use the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API to view or set Object Lock information. For general information about S3 Object Lock capabilities, see Locking objects with Object Lock.

Important

- After you enable Object Lock on a bucket, you can't disable Object Lock or suspend versioning for that bucket.
- S3 buckets with Object Lock can't be used as destination buckets for server access logs. For more information, see the section called “Logging server access”.

Topics

- Permissions for viewing lock information
- Bypassing governance mode
- Using Object Lock with S3 Replication
- Using Object Lock with encryption
- Using Object Lock with Amazon S3 Inventory
- Managing S3 Lifecycle policies with Object Lock
- Managing delete markers with Object Lock
- Using S3 Storage Lens with Object Lock
- Uploading objects to an Object Lock enabled bucket

- Configuring events and notifications
- Setting limits on retention periods with a bucket policy
- Troubleshooting variable retention

Permissions for viewing lock information

You can programmatically view the Object Lock status of an Amazon S3 object version by using the HeadObject or GetObject operations. Both operations return the retention mode, retain until date, and legal hold status for the specified object version. Additionally, you can view the Object Lock status for multiple objects in your S3 bucket using S3 Inventory.

To view an object version's retention mode and retention period, you must have the `s3:GetObjectRetention` permission. To view an object version's legal hold status, you must have the `s3:GetObjectLegalHold` permission. To view a bucket's default retention configuration, you must have the `s3:GetBucketObjectLockConfiguration` permission. If you make a request for an Object Lock configuration on a bucket that doesn't have S3 Object Lock enabled, Amazon S3 returns an error. For information about setting a bucket default, see Configuring S3 Object Lock.

Bypassing governance mode

If you have the `s3:BypassGovernanceRetention` permission, you can perform operations on object versions that are locked in governance mode as if they were unprotected. These operations include deleting an object version, shortening the retention period, or removing the Object Lock retention period by placing a new `PutObjectRetention` request with empty parameters.

To bypass governance mode, you must explicitly indicate in your request that you want to bypass this mode. To do this, include the `x-amz-bypass-governance-retention:true` header with your `PutObjectRetention` API operation request, or use the equivalent parameter with requests made through the AWS CLI or AWS SDKs. The S3 console automatically applies this header for requests made through the S3 console if you have the `s3:BypassGovernanceRetention` permission.

> **Note**
>
> Bypassing governance mode doesn't affect an object version's legal hold status. If an object version has a legal hold enabled, the legal hold remains and prevents requests to overwrite or delete the object version.

Using Object Lock with S3 Replication

You can use Object Lock with S3 Replication to enable automatic, asynchronous copying of locked objects and their retention metadata, across S3 buckets. This means that for replicated objects, Amazon S3 takes the object lock configuration of the source bucket. In other words, if the source bucket has Object Lock enabled, the destination buckets must also have Object Lock enabled. If an object is directly uploaded to the destination bucket (outside of S3 Replication), it takes the Object Lock set on the destination bucket. When you use replication, objects in a *source bucket* are replicated to one or more *destination buckets*.

To set up replication on a bucket with Object Lock enabled, you can use the S3 console, AWS CLI, Amazon S3 REST API, or AWS SDKs.

> **Note**
>
> To use Object Lock with replication, you must grant two additional permissions on the source S3 bucket in the AWS Identity and Access Management (IAM) role that you use to set up replication. The two additional permissions are `s3:GetObjectRetention` and `s3:GetObjectLegalHold`. If the role has an `s3:Get*` permission statement, that statement satisfies the requirement. For more information, see Setting up permissions for live replication.
>
> For general information about S3 Replication, see Replicating objects within and across Regions.
>
> For examples of setting up S3 Replication, see Examples for configuring live replication.

Amazon S3 replicates variable retention settings—event hold and event hold duration—along with other Object Lock metadata. Both source and destination buckets must have Object Lock enabled. When an event hold is released on the source, Amazon S3 replicates the release to the destination, and the destination sets its own final retain-until-date based on the configured duration and the time of replication.

Using Object Lock with encryption

Amazon S3 encrypts all new objects by default. You can use Object Lock with your encrypted objects. For more information, see Protecting data with encryption.

While Object Lock can help prevent Amazon S3 objects from being deleted or overwritten, it does not protect against losing access to the encryption keys or encryption keys being deleted. For example, if you encrypt your objects with AWS KMS server-side encryption and your AWS KMS key is deleted your objects may become unreadable.

Using Object Lock with Amazon S3 Inventory

You can configure Amazon S3 Inventory to create lists of the objects in an S3 bucket on a defined schedule. You can configure Amazon S3 Inventory to include the following Object Lock metadata for your objects:

- The retain until date
- The retention mode
- The legal hold status
- The event hold status
- The event hold duration

While an event hold is on, the `GetObjectRetention`, `HeadObject`, and `GetObject` operations return the computed retain-until-date. If you specified a retain-until-date, the returned value is not earlier than that date.

For objects with an active event hold, Amazon S3 Inventory reports a dynamic retain-until-date based on when Amazon S3 generates the report. For objects with a released event hold or fixed retention, the retain-until-date is fixed. Inventory reports can take up to 48 hours to generate. Therefore, the retain-until-date reflects the report generation time, not the current time. For the most current value, query the object's retention metadata directly by using `GetObjectRetention`.

For more information, see Cataloging and analyzing your data with S3 Inventory.

Managing S3 Lifecycle policies with Object Lock

Object lifecycle management configurations continue to function normally on protected objects, including placing delete markers. However, a locked version of an object cannot be deleted by a S3

Lifecycle expiration policy. Object Lock is maintained regardless of which storage class the object resides in and throughout S3 Lifecycle transitions between storage classes.

For more information about managing object lifecycles, see Managing the lifecycle of objects.

Managing delete markers with Object Lock

Although you can't delete a protected object version, you can still create a delete marker for that object. Placing a delete marker on an object doesn't delete the object or its object versions. However, it makes Amazon S3 behave in most ways as though the object has been deleted. For more information, see Working with delete markers.

> **Note**
>
> Delete markers are not WORM-protected, regardless of any retention period or legal hold in place on the underlying object.

Using S3 Storage Lens with Object Lock

To see metrics for Object Lock-enabled storage bytes and object count, you can use Amazon S3 Storage Lens. S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity.

For more information, see Using S3 Storage Lens to protect your data.

For a complete list of metrics, see Amazon S3 Storage Lens metrics glossary.

Uploading objects to an Object Lock enabled bucket

The `Content-MD5` or `x-amz-sdk-checksum-algorithm` header is required for any request to upload an object with a retention period configured using Object Lock. Theses headers are a way to verify the integrity of your object during upload.

When uploading an object with the Amazon S3 console, S3 automatically adds the `Content-MD5` header. You can optionally specify an additional checksum function and checksum value through the console as the `x-amz-sdk-checksum-algorithm` header. If you use the PutObject API you must specify the `Content-MD5` header, the `x-amz-sdk-checksum-algorithm` header, or both to configure the Object Lock retention period.

For more information, see Checking object integrity in Amazon S3.

Configuring events and notifications

You can use Amazon S3 Event Notifications to track access and changes to your Object Lock configurations and data by using AWS CloudTrail. For information about CloudTrail, see What is AWS CloudTrail? in the *AWS CloudTrail User Guide*.

You can also use Amazon CloudWatch to generate alerts based on this data. For information about CloudWatch, see the What is Amazon CloudWatch? in the *Amazon CloudWatch User Guide*.

Amazon S3 publishes an `s3:ObjectRetention:Put` event notification for explicit `PutObjectRetention` calls. This includes setting fixed or variable retention, releasing an event hold, modifying the duration, and extending a retain-until-date.

Setting limits on retention periods with a bucket policy

You can set minimum and maximum allowable retention periods for a bucket by using a bucket policy. The maximum retention period is 100 years.

The following example shows a bucket policy that uses the `s3:object-lock-remaining-retention-days` condition key to set a maximum retention period of 10 days.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "SetRetentionLimits",
    "Statement": [
        {
            "Sid": "SetRetentionPeriod",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:PutObjectRetention"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "NumericGreaterThan": {
                    "s3:object-lock-remaining-retention-days": "10"
                }
```

```
            }
        }
    ]
}
```

> **Note**
>
> If your bucket is the destination bucket for a replication configuration, you can set up minimum and maximum allowable retention periods for object replicas that are created by using replication. To do so, you must allow the `s3:ReplicateObject` action in your bucket policy. For more information about replication permissions, see the section called "Setting up permissions".

For variable retention, you can control event holds and durations with the following condition keys. Amazon S3 evaluates these keys against the values in a request. They don't apply to an event hold or duration that Amazon S3 applies from a bucket's default retention configuration. For more information, see Set or modify a default variable retention period on an S3 bucket. To restrict bucket defaults, deny the `s3:PutBucketObjectLockConfiguration` action.

- `s3:object-lock-event-hold` – Restricts who can set or release event holds. Values: `ON`, `OFF`.
- `s3:object-lock-event-hold-duration-days` – Lets you enforce a minimum or maximum event hold duration, in days.

The following example denies any `PutObjectRetention` request that sets an event hold duration of less than 90 days:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "RequireMinimumEventHoldDuration",
            "Effect": "Deny",
            "Principal": "*",
            "Action": "s3:PutObjectRetention",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "NumericLessThan": {
```

```
                    "s3:object-lock-event-hold-duration-days": "90"
                }
            }
        }
    ]
}
```

The following example denies any request that sets an event hold on this bucket:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyEventHolds",
            "Effect": "Deny",
            "Principal": "*",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectRetention"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "StringEquals": {
                    "s3:object-lock-event-hold": "ON"
                }
            }
        }
    ]
}
```

Note

A bucket policy that uses the `s3:object-lock-remaining-retention-days` condition key denies a `PutObjectRetention` request when the retention period that you specify exceeds your limit. While an event hold is active, Amazon S3 recalculates the retain-until-date on the service side. The date can move past a limit that the policy enforces at request time. If you don't want this behavior, also restrict the `s3:object-lock-event-hold` condition key to control who can enable event holds.

For more information about bucket policies, see the following topics:

- Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*

  For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.
- Object operations
- Bucket policy examples using condition keys

Troubleshooting variable retention

Use the following information to diagnose and resolve common issues with variable retention and event holds.

Request that sets an event hold is rejected

A request that sets or modifies an event hold fails with a `400 Bad Request` response.

Common causes and actions:

- **No retention mode** – An event hold requires a retention mode. Specify either `GOVERNANCE` or `COMPLIANCE`.
- **Missing or duplicate duration** – An event hold that is set to `ON` requires exactly one event hold duration. Specify the duration in either days or years, but not both.
- **Duration specified with an event hold that is set to OFF** – Don't specify an event hold duration when you release an event hold. Amazon S3 computes the final retain-until-date from the duration that is already configured on the object version.
- **No retain-until-date with an event hold that is set to OFF** – When the event hold is off, the object version must have a retain-until-date. Amazon S3 sets this date in one of two ways. It computes the date when you release an active event hold (a transition from `ON` to `OFF`). Or, you provide an explicit retain-until-date, which is fixed retention. To protect an object with variable retention, create it with the event hold set to `ON` and a duration.
- **Duration outside the allowed range** – An event hold duration must be a positive integer, up to a maximum of 36,500 days or 100 years.

Bucket policy doesn't match a duration that is set in years

There is no years-based condition key. When you set a duration in years, Amazon S3 converts it to days for evaluation against the `s3:object-lock-event-hold-duration-days` condition

key. Amazon S3 counts 1 year as 365 days and doesn't count leap years. Express the limits in your bucket policy in days, and account for this conversion. For example, Amazon S3 evaluates a duration of 1 year as 365 days.

Amazon S3 Inventory report shows an outdated retain-until-date

While an event hold is active, Amazon S3 dynamically computes the retain-until-date. Amazon S3 Inventory reports are generated periodically and can be up to 48 hours old. As a result, a report might not show the current computed retain-until-date. For the current value, use `GetObjectRetention` or `HeadObject`.

Configuring S3 Object Lock

With Amazon S3 Object Lock, you can store objects in Amazon S3 general purpose buckets by using a *write-once-read-many* (WORM) model. You can use S3 Object Lock to prevent an object from being deleted or overwritten for a fixed or variable amount of time, or indefinitely. For general information about Object Lock capabilities, see Locking objects with Object Lock.

Before you lock any objects, you must enable S3 Versioning and Object Lock on a general purpose bucket. Afterward, you can set a fixed or variable retention period, a legal hold, or both.

To work with Object Lock, you must have certain permissions. For a list of the permissions related to various Object Lock operations, see the section called “Required permissions”.

> **Important**
>
> - After you enable Object Lock on a bucket, you can't disable Object Lock or suspend versioning for that bucket.
> - S3 buckets with Object Lock can't be used as destination buckets for server access logs. For more information, see the section called “Logging server access”.

Topics

- Enable Object Lock when creating a new S3 general purpose bucket
- Enable Object Lock on an existing S3 bucket
- Set or modify a legal hold on an S3 object
- Set or modify a fixed retention period on an S3 object

- Set or modify a default fixed retention period on an S3 bucket
- Set or modify a variable retention period on an S3 object
- Release an event hold on an S3 object
- Set or modify a default variable retention period on an S3 bucket

Enable Object Lock when creating a new S3 general purpose bucket

You can enable Object Lock when creating a new S3 general purpose bucket by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. Choose **Create bucket**.

   The **Create bucket** page opens.
4. For **Bucket name**, enter a name for your bucket.

   > **Note**
   >
   > After you create a bucket, you can't change its name. For more information about naming buckets, see General purpose bucket naming rules.
5. For **Region**, choose the AWS Region where you want the bucket to reside.
6. Under **Object Ownership**, choose to disable or enable access control lists (ACLs) and control ownership of objects uploaded in your bucket.
7. Under **Block Public Access settings for this bucket**, choose the Block Public Access settings that you want to apply to the bucket.
8. Under **Bucket Versioning**, choose **Enabled**.

   Object Lock works only with versioned buckets.
9. (Optional) Under **Tags**, you can choose to add tags to your bucket. Tags are key-value pairs that are used to categorize storage and allocate costs.
10. Under **Advanced settings**, find **Object Lock** and choose **Enable**.

You must acknowledge that enabling Object Lock will permanently allow objects in this bucket to be locked.

11. Choose **Create bucket**.

Using the AWS CLI

The following `create-bucket` example creates a new S3 bucket named *amzn-s3-demo-bucket1* with Object Lock enabled:

```
aws s3api create-bucket --bucket amzn-s3-demo-bucket1 --object-lock-enabled-for-bucket
```

For more information and examples, see create-bucket in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

Using the REST API

You can use the REST API to create a new S3 bucket with Object Lock enabled. For more information, see CreateBucket in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

For examples of how to enable Object Lock when creating a new S3 bucket with the AWS SDKs, see CreateBucket code examples in the *Amazon S3 API Reference*.

For examples of how to get the current Object Lock configuration with the AWS SDKs, see GetObjectLockConfiguration code examples in the *Amazon S3 API Reference*.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Object Lock scenario code examples in the *Amazon S3 API Reference*.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Enable Object Lock on an existing S3 bucket

You can enable Object Lock for an existing S3 bucket by using the Amazon S3 console, the AWS CLI, AWS SDKs, or Amazon S3 REST API.

Using the S3 console

> **Note**
>
> Object Lock works only with versioned buckets.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want to enable Object Lock on.
4. Choose the **Properties** tab.
5. Under **Properties**, scroll down to the **Object Lock** section, and choose **Edit**.
6. Under **Object Lock**, choose **Enable**.

   You must acknowledge that enabling Object Lock will permanently allow objects in this bucket to be locked.
7. Choose **Save changes**.

Using the AWS CLI

The following `put-object-lock-configuration` example command sets a 50-day Object Lock retention period on a bucket named *`amzn-s3-demo-bucket1`*:

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-
lock-configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

For more information and examples, see put-object-lock-configuration in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

Using the REST API

You can use the Amazon S3 REST API to enable Object Lock on an existing S3 bucket. For more information, see PutObjectLockConfiguration in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

For examples of how to enable Object Lock for an existing S3 bucket with the AWS SDKs, see PutObjectLockConfiguration code examples in the *Amazon S3 API Reference*.

For examples of how to get the current Object Lock configuration with the AWS SDKs, see GetObjectLockConfiguration code examples in the *Amazon S3 API Reference*.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Object Lock scenario code examples in the *Amazon S3 API Reference*.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Set or modify a legal hold on an S3 object

You can set or remove a legal hold on an S3 object by using the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API.

> **Important**
>
> - If you want to set a legal hold on an object, the object's bucket must already have Object Lock enabled.
> - When you PUT an object version that has an explicit individual retention mode and period in a bucket, the object version's individual Object Lock settings override any bucket property retention settings.

For more information, see the section called "Legal holds".

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the object that you want to set or modify a legal hold on.
4. In the **Objects** list, select the object that you want to set or modify a legal hold on.
5. On the **Object properties** page, find the **Object Lock legal hold** section, and choose **Edit**.
6. Choose **Enable** to set a legal hold or **Disable** to remove a legal hold.
7. Choose **Save changes**.

Using the AWS CLI

The following `put-object-legal-hold` example sets a legal hold on the object *`my-image.fs`* in the bucket named *`amzn-s3-demo-bucket1`*:

```
aws s3api put-object-legal-hold --bucket amzn-s3-demo-bucket1 --key my-image.fs --
legal-hold="Status=ON"
```

The following `put-object-legal-hold` example removes a legal hold on the object *`my-image.fs`* in the bucket named *`amzn-s3-demo-bucket1`*:

```
aws s3api put-object-legal-hold --bucket amzn-s3-demo-bucket1 --key my-image.fs --
legal-hold="Status=OFF"
```

For more information and examples, see put-object-legal-hold in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

Using the REST API

You can use the REST API to set or modify a legal hold on an object. For more information, see PutObjectLegalHold in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

For examples of how to set a legal hold on an object with the AWS SDKs, see PutObjectLegalHold code examples in the *Amazon S3 API Reference*.

For examples of how to get the current legal hold status with the AWS SDKs, see GetObjectLegalHoldConfiguration code examples in the *Amazon S3 API Reference*.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Object Lock scenario code examples in the *Amazon S3 API Reference*.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Set or modify a fixed retention period on an S3 object

You can set or modify a retention period on an S3 object by using the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API.

Important

- If you want to set a retention period on an object, the object's bucket must already have Object Lock enabled.
- When you PUT an object version that has an explicit individual retention mode and period in a bucket, the object version's individual Object Lock settings override any bucket property retention settings.
- The only way to delete an object under the compliance mode before its retention date expires is to delete the associated AWS account.

For more information, see Retention periods.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the object that you want to set or modify a retention period on.
4. In the **Objects** list, select the object that you want to set or modify a retention period on.
5. On the **Object properties** page, find the **Object Lock retention** section, and choose **Edit**.
6. Under **Retention**, choose **Enable** to set a retention period or **Disable** to remove a retention period.
7. If you chose **Enable**, under **Retention mode**, choose either **Governance mode** or **Compliance mode**. For more information, see Retention modes.
8. Under **Retain until date**, choose the date that you want to have the retention period end on. During this period, your object is WORM-protected and can't be overwritten or deleted. For more information, see Retention periods.
9. Choose **Save changes**.

Using the AWS CLI

The following `put-object-retention` example sets a retention period on the object *my-image.fs* in the bucket named *amzn-s3-demo-bucket1* until January 1, 2025:

```
aws s3api put-object-retention --bucket amzn-s3-demo-bucket1 --key my-image.fs --
retention='{ "Mode": "GOVERNANCE", "RetainUntilDate": "2025-01-01T00:00:00" }'
```

For more information and examples, see put-object-retention in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

Using the REST API

You can use the REST API to set a retention period on an object. For more information, see PutObjectRetention in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

For examples of how to set a retention period on an object with the AWS SDKs, see PutObjectRetention code examples in the *Amazon S3 API Reference*.

For examples of how to get the retention period on an object with the AWS SDKs, see GetObjectLockConfiguration code examples in the *Amazon S3 API Reference*.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Object Lock scenario code examples in the *Amazon S3 API Reference*.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Set or modify a default fixed retention period on an S3 bucket

You can set or modify a default retention period on an S3 bucket by using the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API. You specify a duration, in either days or years, for how long to protect every object version placed in the bucket.

Important

- If you want to set a default retention period on a bucket, the bucket must already have Object Lock enabled.
- When you PUT an object version that has an explicit individual retention mode and period in a bucket, the object version's individual Object Lock settings override any bucket property retention settings.
- The only way to delete an object under the compliance mode before its retention date expires is to delete the associated AWS account.

For more information, see Retention periods.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.

3. In the **Buckets** list, choose the name of the bucket that you want to set or modify a default retention period on.
4. Choose the **Properties** tab.
5. Under **Properties**, scroll down to the **Object Lock** section, and choose **Edit**.
6. Under **Default retention**, choose **Enable** to set a default retention or **Disable** to remove a default retention.
7. If you chose **Enable**, under **Retention mode**, choose either **Governance mode** or **Compliance mode**. For more information, see Retention modes.
8. Under **Default retention period**, choose the number of days or years that you want the retention period to last for. Objects placed in this bucket will be locked for this number of days or years. For more information, see Retention periods.
9. Choose **Save changes**.

Using the AWS CLI

The following `put-object-lock-configuration` example command sets a 50-day Object Lock retention period on the bucket named *amzn-s3-demo-bucket1* by using compliance mode:

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-
lock-configuration='{ "ObjectLockEnabled": "Enabled", "Rule": { "DefaultRetention":
 { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

The following `put-object-lock-configuration` example removes the default retention configuration on a bucket:

```
aws s3api put-object-lock-configuration --bucket amzn-s3-demo-bucket1 --object-lock-
configuration='{ "ObjectLockEnabled": "Enabled"}'
```

For more information and examples, see put-object-lock-configuration in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from

the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

Using the REST API

You can use the REST API to set a default retention period on an existing S3 bucket. For more information, see PutObjectLockConfiguration in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

For examples of how to set a default retention period on an existing S3 bucket with the AWS SDKs, see PutObjectLockConfiguration code examples in the *Amazon S3 API Reference*.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Object Lock scenario code examples in the *Amazon S3 API Reference*.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Set or modify a variable retention period on an S3 object

You can set variable retention on an S3 object version by using the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API. The object's bucket must already have Object Lock enabled.

You specify the event hold duration in either days or years, but not both. You can specify 1 to 36,500 days, or 1 to 100 years.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**, and choose the bucket that contains the object.
3. In the **Objects** list, choose the object.
4. On the **Object properties** page, under **S3 Object Lock retention**, choose **Edit**.
5. Choose a **Retention mode** (Compliance or Governance).
6. Under **Retention type**, choose **Variable retention with event hold**.

7. For **Event hold duration**, enter the number of days or years.
8. (Optional) To set a minimum retain-until-date, select **Set an explicit Retain until date in addition to the event hold**, and choose a **Retain until date**. The object stays protected until at least this date even if the event hold is released earlier.
9. Choose **Save changes**.

Using the AWS CLI

The following `put-object-retention` example sets variable retention with Compliance mode and a 30-day duration on the object `my-document.pdf` in the bucket named *amzn-s3-demo-bucket1*:

```
aws s3api put-object-retention \
  --bucket amzn-s3-demo-bucket1 \
  --key my-document.pdf \
  --retention '{"Mode":"COMPLIANCE","EventHold":"ON","EventHoldDuration":{"Days":30}}'
```

The following `put-object-retention` example sets variable retention with a minimum retain-until-date:

```
aws s3api put-object-retention \
  --bucket amzn-s3-demo-bucket1 \
  --key my-document.pdf \
  --retention '{"Mode":"COMPLIANCE","EventHold":"ON","EventHoldDuration":
{"Days":30},"RetainUntilDate":"2027-01-01T00:00:00Z"}'
```

The following `put-object` example sets variable retention at upload time:

```
aws s3api put-object \
  --bucket amzn-s3-demo-bucket1 \
  --key my-document.pdf \
  --body my-document.pdf \
  --object-lock-mode COMPLIANCE \
  --object-lock-event-hold ON \
  --object-lock-event-hold-duration-days 30
```

For more information and examples, see put-object-retention in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

Using the REST API

You can use the REST API to set variable retention on an object. For more information, see PutObjectRetention in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

For examples of how to set variable retention on an object with the AWS SDKs, see PutObjectRetention code examples in the *Amazon S3 API Reference*.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Object Lock scenario code examples in the *Amazon S3 API Reference*.

Release an event hold on an S3 object

You can release an event hold on an S3 object by using the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API. The object's bucket must already have Object Lock enabled.

When you release an event hold, Amazon S3 sets the final retain-until-date to the release time plus the configured duration. The object remains WORM-protected for the full duration after release.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**, and choose the bucket that contains the object.
3. In the **Objects** list, choose the object.
4. On the **Object properties** page, under **S3 Object Lock retention**, choose **Edit**.
5. Under **Event hold**, choose **Turn off event hold** and confirm.

Alternatively, if you want to release the event hold and set a later retain-until-date, under **Retention type** choose **Fixed retention**, specify a retain-until-date, and choose **Save changes**.

Using the AWS CLI

The following `put-object-retention` example releases the event hold on the object `my-document.pdf`:

```
aws s3api put-object-retention \
  --bucket amzn-s3-demo-bucket1 \
  --key my-document.pdf \
  --retention '{"Mode":"COMPLIANCE","EventHold":"OFF"}'
```

For more information and examples, see put-object-retention in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

Using the REST API

You can use the REST API to release an event hold on an object. For more information, see PutObjectRetention in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

For examples of how to release an event hold on an object with the AWS SDKs, see PutObjectRetention code examples in the *Amazon S3 API Reference*.

Set or modify a default variable retention period on an S3 bucket

You can set or modify a default variable retention period on an S3 bucket by using the Amazon S3 console, AWS CLI, AWS SDKs, or Amazon S3 REST API. You specify a duration, in either days or years, for how long to protect every object version placed in the bucket after the event hold is released.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you want to set or modify a default retention period on.
4. Choose the **Properties** tab.
5. Under **Properties**, scroll down to the **Object Lock** section, and choose **Edit**.
6. Under **Default retention**, choose **Enable**.
7. Under **Retention mode**, choose either **Governance mode** or **Compliance mode**.
8. Under **Retention type**, choose **Variable retention with event hold**.
9. Under **Event hold duration**, enter the number of days or years. New objects uploaded without explicit retention settings receive this default mode, event hold, and duration.
10. (Optional) To set a minimum retention period for new objects, select **Add extra retention protection**, and enter a **Default retention period**. Each new object stays protected for at least this period, even if its event hold is released earlier.
11. Choose **Save changes**.

Using the AWS CLI

The following `put-object-lock-configuration` example configures a bucket default with Compliance mode, a default event hold duration of 90 days, and a default retention period of 365 days:

```
aws s3api put-object-lock-configuration \
  --bucket amzn-s3-demo-bucket1 \
  --object-lock-configuration '{
    "ObjectLockEnabled": "Enabled",
    "Rule": {
      "DefaultRetention": {
        "Mode": "COMPLIANCE",
        "Days": 365,
        "DefaultEventHold": {"Days": 90}
      }
    }
```

```
}'
```

For more information and examples, see put-object-lock-configuration in the *AWS CLI Command Reference*.

> **Note**
>
> You can run AWS CLI commands from the console by using AWS CloudShell. AWS CloudShell is a browser-based, pre-authenticated shell that you can launch directly from the AWS Management Console. For more information, see What is CloudShell? in the *AWS CloudShell User Guide*.

Using the REST API

You can use the REST API to set a default variable retention period on an existing S3 bucket. For more information, see PutObjectLockConfiguration in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

For examples of how to set a default retention period on an existing S3 bucket with the AWS SDKs, see PutObjectLockConfiguration code examples in the *Amazon S3 API Reference*.

For an interactive scenario demonstrating different Object Lock features using the AWS SDKs, see Object Lock scenario code examples in the *Amazon S3 API Reference*.

For general information about using different AWS SDKs, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.

Backing up your Amazon S3 data

Amazon S3 is natively integrated with AWS Backup, a fully managed, policy-based service that you can use to centrally define backup policies to protect your data in Amazon S3. After you define your backup policies and assign Amazon S3 resources to the policies, AWS Backup automates the creation of Amazon S3 backups and securely stores the backups in an encrypted backup vault that you designate in your backup plan.

When using AWS Backup for Amazon S3, you can perform the following actions:

- Create continuous backups and periodic backups. Continuous backups are useful for point-in-time restore, and periodic backups are useful to meet your long-term data-retention needs.
- Automate backup scheduling and retention by centrally configuring backup policies.
- Restore backups of Amazon S3 data to a point in time that you specify.
- Access backup data directly through S3 access points without initiating a restore, enabling targeted file recovery, data validation, and compliance auditing.

Along with AWS Backup, you can use S3 Versioning and S3 Replication to help recover from accidental deletions and perform your own self-recovery operations.

Accessing backup data

AWS Backup lets you read Amazon S3 backup data directly through S3 access points without initiating a full restore. You create a backup access point for a specific S3 recovery point, and AWS Backup provisions an S3 access point that you can use with standard S3 read operations such as `GetObject`, `HeadObject`, and `ListObjectsV2`.

For more information, see Backup access points in the *AWS Backup Developer Guide*.

Prerequisites

You must activate S3 Versioning on your bucket before AWS Backup can back it up.

> **Note**
>
> We recommend that you set a lifecycle expiration rule for versioning-enabled buckets that are being backed up. If you do not set a lifecycle expiration period, your Amazon S3 storage costs might increase because AWS Backup retains all versions of your Amazon S3 data.

Getting started

To get started with AWS Backup for Amazon S3, see Creating Amazon S3 backups in the *AWS Backup Developer Guide*.

Restrictions and limitations

To learn about the limitations, see Creating Amazon S3 backups in the *AWS Backup Developer Guide*.

Cost optimization

Amazon S3 offers a range of features and storage classes to help you optimize costs throughout your data lifecycle. Storage classes offer the flexibility to manage your costs, by providing different data-access levels at corresponding costs, with no upfront fees or commitment to how much content you store. Like other AWS services, you pay as you go and pay only for what you use.

Amazon S3 storage classes are purpose-built to provide the lowest cost storage for different access patterns. These include:

- S3 Standard for general-purpose storage of frequently accessed data.
- Amazon S3 Express One Zone for high-performance frequently accessed data in a single-Availability Zone.
- S3 Intelligent-Tiering to automatically optimize costs for data with unknown or changing access patterns.
- S3 Standard-IA (S3 Standard-IA) and S3 One Zone-IA (S3 One Zone-IA) for long-lived, but less frequently accessed data.
- S3 Glacier Instant Retrieval for archive data that needs immediate access.
- S3 Glacier Flexible Retrieval for archive data that doesn't require immediate access but needs the flexibility to retrieve large sets of data at no cost.
- S3 Glacier Deep Archive for long-term archive and digital preservation at the lowest storage costs in the cloud.

You can move objects to the most cost-effective storage class at any time. Additionally, Amazon S3 provides features to manage your data lifecycle. For example, you can use S3 Lifecycle configuration to automate transitioning objects to more cost-effective storage classes, or to automatically delete expired objects based on the rules that you define.

Features such as S3 Storage Class Analysis, cost allocation tagging, and billing and usage reports help you analyze your cost and usage patterns.

Topics

- Billing and usage reporting for Amazon S3
- Understanding and managing Amazon S3 storage classes
- Managing the lifecycle of objects

Billing and usage reporting for Amazon S3

When using Amazon S3, you don't have to pay any upfront fees or commit to how much content you'll store. Like other AWS services, you pay as you go and pay only for what you use.

AWS provides the following reports for Amazon S3:

- **Billing reports** – Multiple reports that provide high-level views of all of the activity for the AWS services that you're using, including Amazon S3. AWS always bills the owner of the S3 bucket for Amazon S3 fees, unless the bucket was created as a Requester Pays bucket. For more information about Requester Pays, see Using Requester Pays general purpose buckets for storage transfers and usage. For more information about billing reports, see AWS Billing reports for Amazon S3.
- **Usage report** – A summary of activity for a specific service, aggregated by hour, day, or month. You can choose which usage type and operation to include. You can also choose how the data is aggregated. For more information, see AWS usage reports for Amazon S3.

The following topics provide information about billing and usage reporting for Amazon S3.

Topics

- Using cost allocation S3 bucket tags
- AWS Billing reports for Amazon S3
- AWS usage reports for Amazon S3
- Understanding your AWS billing and usage reports for Amazon S3
- Billing for Amazon S3 error responses

Using cost allocation S3 bucket tags

To track the storage cost or other criteria for individual projects or groups of projects, label your Amazon S3 buckets using cost allocation tags. A *cost allocation tag* is a key-value pair that you associate with an S3 bucket. After you activate cost allocation tags, AWS uses the tags to organize your resource costs on your cost allocation report. Cost allocation tags can only be used to label buckets. For information about tags used for labeling objects, see Tagging your objects.

The *cost allocation report* lists the AWS usage for your account by product category and linked account user. The report contains the same line items as the detailed billing report (see

Understanding your AWS billing and usage reports for Amazon S3) and additional columns for your tag keys.

AWS provides two types of cost allocation tags, an AWS-generated tag and user-defined tags. AWS defines, creates, and applies the AWS-generated `createdBy` tag for you after an Amazon S3 CreateBucket event. You define, create, and apply *user-defined* tags to your S3 bucket.

You must activate both types of tags separately in the Billing and Cost Management console before they can appear in your billing reports. For more information about AWS-generated tags, see AWS-Generated Cost Allocation Tags.

For more information about activating tags, see Using cost allocation tags in the *AWS Billing User Guide*.

User-defined cost allocation tags

A user-defined cost allocation tag has the following components:

- The tag key. The tag key is the name of the tag. For example, in the tag project/Trinity, project is the key. The tag key is a case-sensitive string that can contain 1 to 128 Unicode characters.

- The tag value. The tag value is a required string. For example, in the tag project/Trinity, Trinity is the value. The tag value is a case-sensitive string that can contain from 0 to 256 Unicode characters.

For details on the allowed characters for user-defined tags and other restrictions, see User-Defined Tag Restrictions in the *AWS Billing User Guide*. For more information about user-defined tags, see User-Defined Cost Allocation Tags in the *AWS Billing User Guide*.

S3 bucket tags

Each S3 bucket has a tag set. A *tag set* contains all of the tags that are assigned to that bucket. A tag set can contain as many as 50 tags, or it can be empty. Keys must be unique within a tag set, but values in a tag set don't have to be unique. For example, you can have the same value in tag sets named project/Trinity and cost-center/Trinity.

Within a bucket, if you add a tag that has the same key as an existing tag, the new value overwrites the old value.

AWS doesn't apply any semantic meaning to your tags. We interpret tags strictly as character strings.

To add, list, edit, or delete tags, you can use the Amazon S3 console, the AWS Command Line Interface (AWS CLI), or the Amazon S3 API.

Managing tags for general purpose buckets

You can add or manage tags for general purpose buckets using the Amazon S3 Console, the AWS Command Line Interface (CLI), the AWS SDKs, or using the S3 APIs. For more information, see the following.

Using the Amazon S3 Console

To create tags in the console, see:

- Viewing the properties for an S3 general purpose bucket.

Using the API

To manage tags using the Amazon S3 API, see the following API pages in the *Amazon Simple Storage Service API Reference*.

- PutBucketTagging
- GetBucketTagging
- DeleteBucketTagging

> **Note**
>
> After you enable ABAC, the `PutBucketTagging` or `DeleteBucketTagging` operations will no longer work. Instead, use the `TagResource` and `UntagResource` APIs to add or delete tags on your buckets. We recommend you use the `TagResource` and `UntagResource` APIs to manage tagging before enabling ABAC on your buckets. To learn more about ABAC, seeEnabling ABAC in general purpose buckets.

Using the CLI

To manage tags using the AWS CLI, see the following pages in the AWS CLI Command Reference.

- put-bucket-tagging
- get-bucket-tagging
- delete-bucket-tagging

More Info

- Using Cost Allocation Tags in the *AWS Billing User Guide*
- Understanding your AWS billing and usage reports for Amazon S3
- AWS Billing reports for Amazon S3

AWS Billing reports for Amazon S3

Your monthly bill from AWS separates your usage information and cost by AWS service and function. There are several AWS Billing reports available: the monthly report, the cost allocation report, and detailed billing reports. For information about how to see your billing reports, see Viewing Your Bill in the *AWS Billing User Guide*.

To track your AWS usage and provide estimated charges associated with your account, you can set up AWS Cost and Usage Reports. For more information, see What are AWSCost and Usage Reports? in the *AWS Data Exports Guide*.

You can also download a usage report that gives more detail about your Amazon S3 storage usage than the billing reports. For more information, see AWS usage reports for Amazon S3.

The following table lists the charges associated with Amazon S3 usage.

Charge	Comments
Storage	You pay for storing objects in your S3 buckets. The rate you're charged depends on your objects' size, how long you stored the objects during the month, and the storage class. Amazon S3 offers the following storage classes: S3 Standard, S3 Express One Zone, S3 Intelligent-Tiering, S3 Standard-IA (IA for infrequent access), S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible

Charge	Comments
	Retrieval, S3 Glacier Deep Archive, or Reduced Redundancy Storage (RRS). For more information about storage classes, see Understanding and managing Amazon S3 storage classes. Be aware that if you have S3 Versioning enabled, you're charged for each version of an object that is retained. For more information about versioning, see How S3 Versioning works.
General purpose buckets	You're not billed for the first 2000 general purpose buckets that you create in your account. However, there is a per-bucket rate for each bucket that you create beyond the first 2000. This rate is billed per bucket/month. For information about general purpose bucket pricing, see Amazon S3 Pricing.
Monitoring and automation	You pay a monthly monitoring and automation fee per object stored in the S3 Intelligent-Tiering storage class to monitor access patterns and move objects between access tiers in S3 Intelligent-Tiering.
Requests	You pay for requests, for example, GET requests, made against your S3 buckets and objects. This includes lifecycle requests. The rates for requests depend on what kind of request you're making. For information about request pricing, see Amazon S3 Pricing.
Retrievals	You pay for retrieving objects that are stored in S3 Standard-IA, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive storage.

Charge	Comments
Early deletes	If you delete an object stored in S3 Standard-IA, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage before the minimum storage commitment has passed, you pay an early deletion fee for that object.
Storage management	You pay for the storage management features (Amazon S3 Inventory, analytics, and object tagging) that are enabled on your account's buckets.
Bandwidth	You pay for all bandwidth into and out of Amazon S3, except for the following: • Data transferred in from the internet • Data transferred out to an Amazon Elastic Compute Cloud (Amazon EC2) instance, when the instance is in the same AWS Region as the S3 bucket • Data transferred out to Amazon CloudFront (CloudFront) You also pay a fee for any data transferred by using Amazon S3 Transfer Acceleration.

For detailed information about Amazon S3 usage charges for storage, data transfer, and services, see Amazon S3 Pricing and the Amazon S3 FAQs.

For information about understanding the codes and abbreviations used in the billing and usage reports for Amazon S3, see Understanding your AWS billing and usage reports for Amazon S3.

More info

- AWS usage reports for Amazon S3
- Using cost allocation S3 bucket tags
- AWS Billing and Cost Management
- Amazon S3 Pricing

AWS usage reports for Amazon S3

When you download a usage report, you can choose to aggregate usage data by hour, day, or month. The Amazon S3 usage report lists operations by usage type and AWS Region. For more detailed reports about your Amazon S3 storage usage, download dynamically generated AWS usage reports. You can choose which usage type, operation, and time period to include. You can also choose how the data is aggregated. For more information about usage reports, see AWS Usage Reportin the *AWS Data Exports User Guide*.

The Amazon S3 usage report includes the following information:

- **Service** – Amazon S3
- **Operation** – The operation performed on your bucket or object. For a detailed explanation of Amazon S3 operations, see Tracking Operations in Your Usage Reports.
- **UsageType** – One of the following values:
  - A code that identifies the type of storage
  - A code that identifies the type of request
  - A code that identifies the type of retrieval
  - A code that identifies the type of data transfer
  - A code that identifies early deletions from S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-Infrequent Access (S3 One Zone-IA), S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage
  - `StorageObjectCount` – The count of objects stored within a given bucket

  For a detailed explanation of Amazon S3 usage types, see Understanding your AWS billing and usage reports for Amazon S3.
- **Resource** – The name of the bucket or table associated with the listed usage.
- **StartTime** – Start time of the day that the usage applies to, in Coordinated Universal Time (UTC).

- **EndTime** – End time of the day that the usage applies to, in Coordinated Universal Time (UTC).
- **UsageValue** – One of the following volume values. The typical unit of measurement for data is gigabytes (GB). However, depending on the service and the report, terabytes (TB) might appear instead.
  - The number of requests during the specified time period
  - The amount of data transferred
  - The amount of data stored in a given hour
  - The amount of data associated with restorations from S3 Standard-IA, S3 One Zone-IA, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage

> **Tip**
>
> For detailed information about every request that Amazon S3 receives for your objects, turn on server access logging for your buckets. For more information, see Logging requests with server access logging.

You can download a usage report as an XML or a comma-separated values (CSV) file. The following is an example CSV usage report opened in a spreadsheet application.

Service	Operation	UsageType	Resource	StartTime	EndTime	UsageValue
AmazonS3	HeadBucket	USW2-C3DataTransfer-Out-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	15309
AmazonS3	PutObject	USW2-C3DataTransfer-In-Bytes	admin-created3	6/1/2017 0:00	7/1/2017 0:00	19062
AmazonS3	HeadBucket	USW2-Requests-Tier2	admin-created3	6/1/2017 0:00	7/1/2017 0:00	68
AmazonS3	PutObjectForRepl	USW1-Requests-SIA-Tier1	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	178294
AmazonS3	PutObjectForRepl	USW1-USW2-AWS-In-Bytes	ca-example-bucket	6/1/2017 0:00	7/1/2017 0:00	387929083
AmazonS3	GetObjectForRepl	USW2-Requests-NoCharge	admin-created3	6/1/2017 0:00	7/1/2017 0:00	108
AmazonS3	GetObjectForRepl	USW2-USW1-AWS-Out-Bytes	my-test-bucket-bash	6/1/2017 0:00	7/1/2017 0:00	387910021

For more information, see Understanding your AWS billing and usage reports for Amazon S3.

Downloading the AWS Usage Report

You can download a usage report as an XML or a CSV file.

To download the usage report

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the title bar, choose your username or account ID, and then choose **Billing and Cost Management**.
3. In the navigation pane, choose **Legacy Pages** and choose **Cost and usage reports**.
4. Under AWS Usage Report, choose **Create a Usage Report**.
5. On the **Download usage report** page, choose the following settings:
   - **Services** – Choose **Amazon Simple Storage Service**.
   - **Usage Types** – For a detailed explanation of Amazon S3 usage types, see Understanding your AWS billing and usage reports for Amazon S3.
   - **Operation** – For a detailed explanation of Amazon S3 operations, see Tracking Operations in Your Usage Reports.
   - **Time Period** – The time period that you want the report to cover.
   - **Report Granularity** – Whether you want the report to include subtotals by the hour, by the day, or by the month.
6. Choose **Download**, choose the download format (**XML Report** or **CSV Report**), and then follow the prompts to open or save the report.

More info

- Understanding your AWS billing and usage reports for Amazon S3
- AWS Billing reports for Amazon S3

Understanding your AWS billing and usage reports for Amazon S3

When you use Amazon S3, we include related codes in your AWS billing and usage reports. Reviewing these codes helps you understand your Amazon S3 costs and usage patterns. Tracking and managing your expenses is essential for optimizing your costs. For more information, see Amazon S3 pricing.

Amazon S3 billing and usage reports use codes and abbreviations. For usage types in the table that follows, replace *`region`*, *`region1`*, and *`region2`* with abbreviations from AWS Region billing codes in the *AWS Regions and Availability Zones User Guide*.

> **Note**
>
> For the US East (N. Virginia) Region, the region prefix is omitted from usage type codes. For example, it will show `TimedStorage-ByteHrs` instead of `USE1-TimedStorage-ByteHrs`.

For S3 Multi-Region Access Points usage types in the table that follows, replace *regiongroup1* and *regiongroup2* with abbreviations from this list:

- **AP:** Asia Pacific
- **AU:** Australia
- **EU:** Europe
- **IN:** India
- **NA:** North America
- **SA:** South America

Region groups are geographical groupings of several AWS Regions. For more information, see Regions and Availability Zones. For information about pricing by AWS Region, see Amazon S3 Pricing.

The first column in the following table lists usage types that appear in your billing and usage reports. The typical unit of measurement for data is gigabytes (GB). However, depending on the service and the report, terabytes (TB) might appear instead.

Usage Type	Units	Granularity	Description
region1-*region2*-AWS-In-Bytes	GB	Hourly	The amount of accelerated data transferred to *region1* from *region2*
region1-*region2*-AWS-In-Bytes-T1	GB	Hourly	The amount of T1 accelerated data transferred to *region1* from *region2*, where

Usage Type	Units	Granularity	Description
			T1 refers to CloudFront requests to points of presence (POPs) in the United States, Europe, and Japan
region1-*region2*-AWS-In-ABytes-T2	GB	Hourly	The amount of T2 accelerated data transferred to *region1* from *region2*, where T2 refers to CloudFront requests to POPs in all other AWS edge locations
region1-*region2*-AWS-In-Bytes	GB	Hourly	The amount of data transferred to *region1* from *region2*
region1-*region2*-AWS-Out-ABytes	GB	Hourly	The amount of accelerated data transferred from *region1* to *region2*
region1-*region2*-AWS-Out-ABytes-T1	GB	Hourly	The amount of T1 accelerated data transferred from *region1* to *region2*, where T1 refers to CloudFront requests to POPs in the United States, Europe, and Japan

Usage Type	Units	Granularity	Description
region1-*region2*-AWS-Out-ABytes-T2	GB	Hourly	The amount of T2 accelerated data transferred from *region1* to *region2*, where T2 refers to CloudFront requests to POPs in all other AWS edge locations
region1-*region2*-AWS-Out-Bytes	GB	Hourly	The amount of data transferred from *region1* to *region2*
region-BatchOperations-Jobs	Count	Hourly	The number of S3 Batch Operations jobs performed
region-BatchOperations-Objects	Count	Hourly	The number of object operations performed by S3 Batch Operations
region-Bulk-Retrieval-Bytes	GB	Hourly	The amount of data retrieved with Bulk S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive requests
region-BytesDeleted-GDA	GB	Monthly	The amount of data deleted by a DeleteObject operation from S3 Glacier Deep Archive storage

Usage Type	Units	Granularity	Description
region-BytesDeleted-GIR	GB	Monthly	The amount of data deleted by a DeleteObject operation from S3 Glacier Instant Retrieval storage.
region-BytesDeleted-GLACIER	GB	Monthly	The amount of data deleted by a DeleteObject operation from S3 Glacier Flexible Retrieval storage
region-BytesDeleted-INT	GB	Monthly	The amount of data deleted by a DeleteObject operation from S3 Intelligent-Tiering storage
region-BytesDeleted-RRS	GB	Monthly	The amount of data deleted by a DeleteObject operation from Reduced Redundancy Storage (RRS) storage
region-BytesDeleted-SIA	GB	Monthly	The amount of data deleted by a DeleteObject operation from S3 Standard-IA storage
region-BytesDeleted-STANDARD	GB	Monthly	The amount of data deleted by a DeleteObject operation from S3 Standard storage

Usage Type	Units	Granularity	Description
region-BytesDeleted-ZIA	GB	Monthly	The amount of data deleted by a DeleteObject operation from S3 One Zone-IA storage
region-C3DataTransfer-In-Bytes	GB	Hourly	The amount of data transferred into Amazon S3 from Amazon EC2 within the same AWS Region
region-C3DataTransfer-Out-Bytes	GB	Hourly	The amount of data transferred from Amazon S3 to Amazon EC2 within the same AWS Region
region-CloudFront-In-Bytes	GB	Hourly	The amount of data transferred into an AWS Region from a CloudFront distribution
region-CloudFront-Out-Bytes	GB	Hourly	The amount of data transferred from an AWS Region to a CloudFront distribution
region-DataTransfer-In-Bytes	GB	Hourly	The amount of data transferred into Amazon S3 from the internet
region-DataTransfer-Out-Bytes	GB	Hourly	The amount of data transferred from Amazon S3 to the internet[1]

Usage Type	Units	Granularity	Description
region-DataTransfer-Regional-Bytes	GB	Hourly	The amount of data transferred from Amazon S3 to AWS resources within the same AWS Region
region-EarlyDelete-ByteHrs	GB-Hours	Hourly	Prorated storage usage for objects deleted from, S3 Glacier Flexible Retrieval storage before the 90-day minimum commitment ended[2]
region-EarlyDelete-GDA	GB-Hours	Hourly	Prorated storage usage for objects deleted from S3 Glacier Deep Archive storage before the 180-day minimum commitment ended [2]
region-EarlyDelete-GIR	GB-Hours	Hourly	Prorated storage usage for objects deleted or transitioned from S3 Glacier Instant Retrieval before the 90-day minimum commitment ended.

Usage Type	Units	Granularity	Description
region-EarlyDelete-GIR-SmObjects	GB-Hours	Hourly	Prorated storage usage for small objects (smaller than 128 KB) that were deleted from S3 Glacier Instant Retrieval before the 90-day minimum commitment ended.
region-EarlyDelete-SIA	GB-Hours	Hourly	Prorated storage usage for objects deleted from S3 Standard-IA before the 30-day minimum commitment ended[3]
region-EarlyDelete-SIA-SmObjects	GB-Hours	Hourly	Prorated storage usage for small objects (smaller than 128 KB) that were deleted from S3 Standard-IA before the 30-day minimum commitment ended[3]
region-EarlyDelete-ZIA	GB-Hours	Hourly	Prorated storage usage for objects deleted from S3 One Zone-IA before the 30-day minimum commitment ended[3]

Usage Type	Units	Granularity	Description
region-EarlyDelete-ZIA-SmObjects	GB-Hours	Hourly	Prorated storage usage for small objects (smaller than 128 KB) that were deleted from S3 One Zone-IA before the 30-day minimum commitment ended[3]
region-Expedited-Retrieval-Bytes	GB	Hourly	The amount of data retrieved with Expedited S3 Glacier Flexible Retrieval requests
Global-Bucket-Hrs-FreeTier	Bucket	Monthly	The number of general purpose buckets in your account within the 2000 bucket account-level free tier
Global-Bucket-Hrs	Bucket	Monthly	The number of general purpose buckets in your account beyond the 2000 bucket account-level free tier
region-Inventory-Objects Listed	Objects	Hourly	The number of objects listed for an object group (objects are grouped by bucket or prefix) with an inventory list
region-Metadata-Updates	Updates	Hourly	Per update fee for updates processed by S3 Metadata

Usage Type	Units	Granularity	Description
region-Metadata-Annotation-Processed-Bytes	GB	Hourly	Data processed by the S3 Metadata annotation table
region-Monitoring-Automation-INT	Objects	Hourly	The number of unique objects monitored and auto-tiered in the S3 Intelligent-Tiering storage class
region-MRAP-Out-Bytes	GB	Hourly	The amount of data transferred through an S3 Multi-Region Access Points endpoint out of buckets in a Region (MRAP data routing pricing).
region-MRAP-In-Bytes	GB	Hourly	The amount of data transferred through an S3 Multi-Region Access Points endpoint out of buckets in a Region (MRAP data routing pricing).

Usage Type	Units	Granularity	Description
regiongroup1-regiongroup2- -MRAP-Out-Bytes	GB	Hourly	The amount of data transferred through an S3 Multi-Region Access Points endpoint from a bucket in *regiongroup1* to a client in *regiongroup2* located outside of the AWS network.
regiongroup1-regiongroup2- -MRAP-In-Bytes	GB	Hourly	The amount of data transferred through an S3 Multi-Region Access Points endpoint to a bucket in *regiongroup1* from a client in *regiongroup2* located outside of the AWS network.
region-OverwriteBytes-Copy-GDA	GB	Monthly	The amount of data overwritten by a CopyObject operation from S3 Glacier Deep Archive storage
region-OverwriteBytes-Copy-GIR	GB	Monthly	The amount of data overwritten by a CopyObject operation from S3 Glacier Instant Retrieval storage.

Usage Type	Units	Granularity	Description
region-OverwriteBytes-Copy-GLACIER	GB	Monthly	The amount of data overwritten by a CopyObject operation from S3 Glacier Flexible Retrieval storage
region-OverwriteBytes-Copy-INT	GB	Monthly	The amount of data overwritten by a CopyObject operation from S3 Intelligent-Tiering storage
region-OverwriteBytes-Copy-RRS	GB	Monthly	The amount of data overwritten by a CopyObject operation from Reduced Redundancy Storage (RRS) storage
region-OverwriteBytes-Copy-SIA	GB	Monthly	The amount of data overwritten by a CopyObject operation from S3 Standard-IA storage
region-OverwriteBytes-Copy-STANDARD	GB	Monthly	The amount of data overwritten by a CopyObject operation from S3 Standard storage

Usage Type	Units	Granularity	Description
region-OverwriteBytes-Copy-ZIA	GB	Monthly	The amount of data overwritten by a CopyObject operation from S3 One Zone-IA storage
region-OverwriteBytes-Put-GDA	GB	Monthly	The amount of data overwritten by a PutObject operation from S3 Glacier Deep Archive storage
region-OverwriteBytes-Put-GIR	GB	Monthly	The amount of data overwritten by a PutObject operation from S3 Glacier Instant Retrieval storage.
region-OverwriteBytes-Put-GLACIER	GB	Monthly	The amount of data overwritten by a PutObject operation from S3 Glacier Flexible Retrieval storage
region-OverwriteBytes-Put-INT	GB	Monthly	The amount of data overwritten by a PutObject operation from S3 Intelligent-Tiering storage

Usage Type	Units	Granularity	Description
region-OverwriteBytes-Put-RRS	GB	Monthly	The amount of data overwritten by a PutObject operation from Reduced Redundancy Storage (RRS) storage
region-OverwriteBytes-Put-SIA	GB	Monthly	The amount of data overwritten by a PutObject operation from S3 Standard-IA storage
region-OverwriteBytes-Put-STANDARD	GB	Monthly	The amount of data overwritten by a PutObject operation from S3 Standard storage
region-OverwriteBytes-Put-ZIA	GB	Monthly	The amount of data overwritten by a PutObject operation from S3 One Zone-IA storage

Usage Type	Units	Granularity	Description
region1-*region2*-S3RTC-In-Bytes	GB	Monthly	The amount of data transferred for S3 Replication Time Control (S3 RTC) from *region2* to *region1* by the PutObjectReplTime , GetObjectReplTime , InitiateMultipartUploadReplTime , UploadPartReplTime , CompleteMultipartUploadReplTime , and WriteACLReplTime operations
region1-*region2*-S3RTC-Out-Bytes	GB	Monthly	The amount of data transferred for S3 Replication Time Control (S3 RTC) from *region1* to *region2* by the PutObjectReplTime , GetObjectReplTime , InitiateMultipartUploadReplTime , UploadPartReplTime , CompleteMultipartUploadReplTime , and WriteACLReplTime operations

Usage Type	Units	Granularity	Description
region-Annotation-TimedStorage-ByteHrs	GB-Month	Daily	Storage for annotations, billed at the Amazon S3 Standard rate
region-Requests-Annotation-Tier1	Count	Hourly	The number of PUT, COPY, POST, and LIST requests for annotations (PutObjectAnnotation , ListObjectAnnotations). Billed per 1,000 requests.
region-Requests-Annotation-Tier2	Count	Hourly	The number of GET requests for annotations (GetObjectAnnotation). Billed per 1,000 requests.
region-Requests-GDA-Tier1	Count	Hourly	The number of PUT, COPY, POST, CreateMultipartUpload , UploadPart , or CompleteMultipartUpload requests on S3 Glacier Deep Archive objects [6]
region-Requests-GDA-Tier2	Count	Hourly	The number of GET and HEAD requests on S3 Glacier Deep Archive objects

Usage Type	Units	Granularity	Description
region-Requests-GDA-Tier3	Count	Hourly	The number of S3 Glacier Deep Archive standard restore requests
region-Requests-GDA-Tier5	Count	Hourly	The number of Bulk S3 Glacier Deep Archive restore requests
region-Requests-GIR-Tier1	Count	Hourly	The number of PUT, COPY, or POST requests on S3 Glacier Instant Retrieval objects.
region-Requests-GIR-Tier2	Count	Hourly	The number of GET and all other non-S3 Glacier Instant Retrieval-Tier1 requests on S3 Glacier Instant Retrieval objects.
region-Requests-GLACIER-Tier1	Count	Hourly	The number of PUT, COPY, POST, CreateMultipartUpload , UploadPart , or CompleteMultipartUpload requests on S3 Glacier Flexible Retrieval objects [6]
region-Requests-GLACIER-Tier2	Count	Hourly	The number of GET and all other requests not listed on S3 Glacier Flexible Retrieval objects

Usage Type	Units	Granularity	Description
region-Requests-INT-Tier1	Count	Hourly	The number of PUT, COPY, or POST requests on S3 Intelligent-Tiering objects
region-Requests-INT-Tier2	Count	Hourly	The number of GET and all other non-Tier1 requests for S3 Intelligent-Tiering objects
region-Requests-SIA-Tier1	Count	Hourly	The number of PUT, COPY, or POST requests on S3 Standard-IA objects
region-Requests-SIA-Tier2	Count	Hourly	The number of GET and all other non-S3 Glacier Instant Retrieval-Tier1 requests on S3 Standard-IA objects
region-Requests-Tier1	Count	Hourly	The number of PUT, COPY, or POST requests for S3 Standard, RRS, and tags, plus LIST requests for all buckets and objects
region-Requests-Tier2	Count	Hourly	The number of GET and all other non-Tier1 requests

Usage Type	Units	Granularity	Description
region-Requests-Tier3	Count	Hourly	The number of lifecycle requests to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive and standard S3 Glacier Flexible Retrieval restore requests
region-Requests-Tier4	Count	Hourly	The number of lifecycle transitions to S3 Glacier Instant Retrieval, S3 Intelligent-Tiering, S3 Standard-IA, or S3 One Zone-IA storage
region-Requests-Tier5	Count	Hourly	The number of Bulk S3 Glacier Flexible Retrieval restore requests
region-Requests-Tier6	Count	Hourly	The number of Expedited S3 Glacier Flexible Retrieval restore requests
region-Requests-Tier8	Count	Hourly	The number of S3 Access Grants requests
region-Requests-XZ-Tier1	Count	Hourly	The number of PUT or COPY requests on S3 Express One Zone objects

Usage Type	Units	Granularity	Description
region-Requests-XZ-Tier2	Count	Hourly	The number of GET and all other non-S3 Express One Zone-Tier1 requests on S3 Express One Zone objects
region-Requests-ZIA-Tier1	Count	Hourly	The number of PUT, COPY, or POST requests on S3 One Zone-IA objects
region-Requests-ZIA-Tier2	Count	Hourly	The number of GET and all other non-S3 One Zone-IA-Tier1 requests on S3 One Zone-IA objects
region-Retrieval-GIR	GB	Hourly	The amount of data retrieved from S3 Glacier Instant Retrieval storage.
region-Retrieval-SIA	GB	Hourly	The amount of data retrieved from S3 Standard-IA storage
region-Retrieval-XZ	GB	Hourly	The portion of the data that exceeds 512 KB in a given retrieval request (PUT or COPY) with S3 Express One Zone storage
region-Retrieval-ZIA	GB	Hourly	The amount of data retrieved from S3 One Zone-IA storage

Usage Type	Units	Granularity	Description
region-S3DSSE-In-Bytes	GB	Monthly	The amount of data dual-encrypted by Amazon S3
region-S3DSSE-Out-Bytes	GB	Monthly	The amount of dual-encrypted data decrypted by Amazon S3
region-S3G-DataTransfer-In-Bytes	GB	Hourly	The amount of data transferred into Amazon S3 to restore objects from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage
region-S3G-DataTransfer-Out-Bytes	GB	Hourly	The amount of data transferred from Amazon S3 to transition objects to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage
region-Select-Returned-Bytes	GB	Hourly	The amount of data returned with Select requests from S3 Standard storage
region-Select-Returned-GIR-Bytes	GB	Hourly	The amount of data returned with Select requests from S3 Glacier Instant Retrieval storage.

Usage Type	Units	Granularity	Description
region-Select-Returned-INT-Bytes	GB	Hourly	The amount of data returned with Select requests from S3 Intelligent-Tiering storage
region-Select-Returned-SIA-Bytes	GB	Hourly	The amount of data returned with Select requests from S3 Standard-IA storage
region-Select-Returned-ZIA-Bytes	GB	Hourly	The amount of data returned with Select requests from S3 One Zone-IA storage
region-Select-Scanned-Bytes	GB	Hourly	The amount of data scanned with Select requests from S3 Standard storage
region-Select-Scanned-GIR-Bytes	GB	Hourly	The amount of data scanned with Select requests from S3 Glacier Instant Retrieval storage.
region-Select-Scanned-INT-Bytes	GB	Hourly	The amount of data scanned with Select requests from S3 Intelligent-Tiering storage

Usage Type	Units	Granularity	Description
region-Select-Scanned-SIA-Bytes	GB	Hourly	The amount of data scanned with Select requests from S3 Standard-IA storage
region-Select-Scanned-ZIA-Bytes	GB	Hourly	The amount of data scanned with Select requests from S3 One Zone-IA storage
region-Standard-Retrieval-Bytes	GB	Hourly	The amount of data retrieved with standard S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive requests
region-StorageAnalytics-ObjCount	Objects	Hourly	The number of unique objects monitored in each Storage Class Analysis configuration.
region-StorageLens-ObjCount	Objects	Daily	The number of unique objects in each S3 Storage Lens dashboard that are tracked by S3 Storage Lens advanced metrics and recommendations.
region-StorageLensFreeTier-ObjCount	Objects	Daily	The number of unique objects in each S3 Storage Lens dashboard that are tracked by S3 Storage Lens usage metrics.

Usage Type	Units	Granularity	Description
StorageObjectCount	Count	Daily	The number of objects stored within a given bucket
region-Tables-CompactedObjects	Objects	Hourly	The number of objects compacted in Amazon S3 table buckets using the binpack compaction strategy in an Amazon S3 Tables compaction job.
region-Tables-SortCompactedObjects-	Objects	Hourly	The number of objects compacted in Amazon S3 table buckets using sort or z-order compaction strategies in an Amazon S3 Tables compaction job.
region-Tables-MonitoredObjects	Objects	Hourly	The number of objects in Amazon S3 table buckets
region-Tables-ProcessedBytes	GB	Hourly	The amount of data processed for compaction in Amazon S3 table buckets
region-Tables-Requests-Tier1	Count	Hourly	The number of PUT requests on Amazon S3 table buckets

Usage Type	Units	Granularity	Description
region-Tables-Requests-Tier2	Count	Hourly	The number of GET and all other non-Tier1 requests on Amazon S3 table buckets
region-Tables-TimedStorage-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in Amazon S3 table buckets
region-TagStorage-TagHrs	Tag-Hours	Daily	The total of tags on all objects in the bucket reported by hour
region-TimedStorage-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in S3 Standard storage
region-TimedStorage-GDA-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in S3 Glacier Deep Archive storage
region-TimedStorage-GDA-Staging	GB-Month	Daily	The number of GB-months that data was stored in S3 Glacier Deep Archive staging storage
region-TimedStorage-GIR-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in S3 Glacier Instant Retrieval storage.

Usage Type	Units	Granularity	Description
region-TimedStorage-GIR-SmObjects	GB-Month	Daily	The number of GB-months that small objects (smaller than 128 KB) were stored in S3 Glacier Instant Retrieval storage.
region-TimedStorage-GlacierByteHrs	GB-Month	Daily	The number of GB-months that data was stored in S3 Glacier Flexible Retrieval storage
region-TimedStorage-GlacierStaging	GB-Month	Daily	The number of GB-months that data was stored in S3 Glacier Flexible Retrieval staging storage
region-TimedStorage-INT-FA-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in the Frequent Access tier of S3 Intelligent-Tiering storage[5]
region-TimedStorage-INT-IA-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in the Infrequent Access tier of S3 Intelligent-Tiering storage

Usage Type	Units	Granularity	Description
region-TimedStorage-INT-AA-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in the Archive Access tier of S3 Intelligent-Tiering storage
region-TimedStorage-INT-AIA-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in the Archive Instant Access tier of S3 Intelligent-Tiering storage
region-TimedStorage-INT-DAA-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in the Deep Archive Access tier of S3 Intelligent-Tiering storage
region-TimedStorage-RRS-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in Reduced Redundancy Storage (RRS) storage
region-TimedStorage-SIA-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in S3 Standard-IA storage

Usage Type	Units	Granularity	Description
region-TimedStorage-SIA-SmObjects	GB-Month	Daily	The number of GB-months that small objects (smaller than 128 KB) were stored in S3 Standard-IA storage[4]
region-TimedStorage-XZ-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in S3 Express One Zone storage
region-TimedStorage-ZIA-ByteHrs	GB-Month	Daily	The number of GB-months that data was stored in S3 One Zone-IA storage
region-TimedStorage-ZIA-SmObjects	GB-Month	Daily	The number of GB-months that small objects (smaller than 128 KB) were stored in S3 One Zone-IA storage
region-Upload-XZ	GB	Hourly	The amount of data that exceeds 512 KB in a given upload request (PUT or COPY) with S3 Express One Zone

Notes

1. The `Global-Bucket-Hrs` and `Global-Bucket-Hrs-FreeTier` usage types apply to general purpose buckets in commercial AWS Regions and AWS GovCloud (US).
2. If you terminate a transfer before completion, the amount of data that is transferred might exceed the amount of data that your application receives. This discrepancy can occur because

a transfer termination request cannot be executed instantaneously, and some amount of data might be in transit, pending execution of the termination request. This data in transit is billed as data transferred "out."

3. For objects archived to S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage classes that are deleted, overwritten, or transitioned before their minimum storage commitment period (90 days for S3 Glacier Instant Retrieval and S3 Glacier Flexible Retrieval, 180 days for S3 Glacier Deep Archive), a prorated charge per gigabyte applies for the remaining days. These charges are reflected as `EarlyDelete-GIR`, `EarlyDelete-ByteHrs`, and `EarlyDelete-GDA` usage types respectively on your bill.
4. For objects that are in S3 Standard-IA or S3 One Zone-IA storage, when they are deleted, overwritten, or transitioned to a different storage class before 30 days, there is a prorated charge per gigabyte for the remaining days.
5. For small objects (smaller than 128 KB) that are in S3 Standard-IA or S3 One Zone-IA storage, when they are deleted, overwritten, or transitioned to a different storage class before 30 days, there is a prorated charge per gigabyte for the remaining days.
6. There is no minimum billable object size for objects in the S3 Intelligent-Tiering storage class. Objects that are smaller than 128 KB are not monitored or eligible for auto-tiering. Smaller objects are always stored in the S3 Intelligent-Tiering Frequent Access tier.
7. When you initiate a `CreateMultipartUpload`, `UploadPart`, or `UploadPartCopy` request to either the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes, requests are billed at S3 Standard request rates until you complete the multipart upload. After the upload is completed, the single `CompleteMultipartUpload` request is billed at the PUT rate for the destination S3 Glacier storage. In-progress multipart upload parts for a PUT to the S3 Glacier Flexible Retrieval storage class are billed as S3 Glacier Flexible Retrieval Staging Storage at S3 Standard storage rates until the upload is completed. Similarly, in-progress multipart upload parts for a PUT to the S3 Glacier Deep Archive storage class are billed as S3 Glacier Deep Archive Staging Storage at S3 Standard storage rates until the upload is completed.
8. S3 Express One Zone applies a flat per-request charge for request sizes up to 512 KB. An additional per GB charge is applied for PUT requests and GET requests for the portion of request greater than 512 KB.
9. For information about supported features for S3 Express One Zone storage class, see Amazon S3 features not supported by directory buckets.
10. Usage types with units that are billed in GB are calculated in bytes in the usage reports.

11A GB-Month is derived by taking the total number of GB-hours, aggregating these over the course of a month, and then dividing by the number of hours in that month. To learn more see, Frequently Asked Questions: How will I be charged and billed for my use of Amazon S3?

> **Note**
>
> In general, S3 bucket owners are billed for requests with HTTP `200 OK` successful responses and HTTP `4XX` client error responses. Bucket owners aren't billed for HTTP `5XX` server error responses, such as HTTP `503 Slow Down` errors. For more information on S3 error codes under HTTP `3XX` and `4XX` status codes that aren't billed, see Billing for Amazon S3 error responses. For more information about billing charges if your bucket is configured as a Requester Pays bucket, see How Requester Pays charges work.

Tracking Operations in Your Usage Reports

Operations describe the action taken on your AWS object or bucket by the specified usage type. Operations are indicated by self-explanatory codes, such as `PutObject` or `ListBucket`. To see which actions on your bucket generated a specific type of usage, use these codes. When you create a usage report, you can choose to include **All Operations**, or a specific operation, for example, `GetObject`, to report on.

More info

- AWS usage reports for Amazon S3
- AWS Billing reports for Amazon S3
- Amazon S3 Pricing
- Amazon S3 FAQs

Billing for Amazon S3 error responses

In general, S3 bucket owners are billed for requests with HTTP `200 OK` successful responses and HTTP `4XX` client error responses. Bucket owners aren't billed for HTTP `5XX` server error responses, such as HTTP `503 Slow Down` errors. For more information about billing charges if your bucket is configured as a Requester Pays bucket, see How Requester Pays charges work.

The following table lists specific error codes under HTTP 3XX and 4XX status codes that aren't billed. For buckets configured with website hosting, applicable request and other charges will still apply when S3 returns a custom error document or for custom redirects.

> **Note**
>
> For `AccessDenied` (HTTP `403 Forbidden`), S3 doesn't charge the bucket owner when the request is initiated outside of the bucket owner's individual AWS account or the bucket owner's AWS organization.

HTTP status code	Error code	Description of error code
301 Moved Permanently	PermanentRedirect	The bucket that you are attempting to access must be addressed using the specified endpoint. Send all future requests to this endpoint.
	PermanentRedirectControlError	The API operation you are attempting to access must be addressed using the specified endpoint. Send all future requests to this endpoint.
307 Temporary Redirect	TemporaryRedirect	You are being redirected to the bucket while the Domain Name System (DNS) server is being updated.

HTTP status code	Error code	Description of error code
400 Bad Request	AuthorizationHeaderMalformed	The authorization header that you provided is not valid.
	AuthorizationQueryParametersError	The authorization query parameters that you provided are not valid.
	ConnectionClosedByRequester	Returned to the original caller when an error is encountered while reading the WriteGetObjectResponse body.
	DeviceNotActiveError	The device is not currently active.
	EndpointNotFound	Direct requests to the correct endpoint.
	ExpiredToken	The provided token has expired.
	IllegalLocationConstraintException	You are trying to access a bucket from a different Region than where the bucket exists. To avoid this error, use the `--region` option. For example: `aws s3 cp awsexample.txt s3://amzn-s3-demo-bucket/ --region ap-east-1`.

HTTP status code	Error code	Description of error code	
	InvalidArgument	This error might occur for the following reasons: • The specified argument was not valid. • The request was missing a required header. • The specified argument was incomplete or in the wrong format. • The specified argument must have a length greater than or equal to 3.	
	InvalidBucketOwnerAWSAccountID	The value of the expected bucket owner parameter must be an AWS account ID.	
	InvalidDigest	The Content-MD5 or checksum value that you specified is not valid.	

HTTP status code	Error code	Description of error code	
	InvalidEncryptionAlgorithmError	The encryption request that you specified is not valid. The valid value is AES256.	
	InvalidHostHeader	The host headers provided in the request used the incorrect style addressing.	
	InvalidHttpMethod	The request is made using an unexpected HTTP method.	

HTTP status code	Error code	Description of error code	
	InvalidRequest	This error might occur for the following reasons: • The request is using the wrong signature version. Use `AWS4-HMAC-SHA256` (Signature Version 4). • An access point can be created only for an existing bucket. • The access point is not in a state where it can be deleted. • An access point can be listed only for an existing bucket. • The next token is not valid. • At least one action must be specified in a lifecycle rule. • At least one lifecycle rule must be specified. •	

HTTP status code	Error code	Description of error code	
		The number of lifecycle rules must not exceed the allowed limit of 1000 rules. • The range for the `MaxResults` parameter is not valid. • SOAP requests must be made over an HTTPS connection. • Amazon S3 Transfer Acceleration is not supported for buckets with non-DNS compliant names. • Amazon S3 Transfer Acceleration is not supported for buckets with periods (.) in their names. • The Amazon S3 Transfer Acceleration endpoint supports only virtual style requests. • Amazon S3 Transfer Acceleration is not	

HTTP status code	Error code	Description of error code	
		configured on this bucket. • Amazon S3 Transfer Acceleration is disabled on this bucket. • Amazon S3 Transfer Acceleration is not supported on this bucket. For assistance, contact Support. • Amazon S3 Transfer Acceleration cannot be enabled on this bucket. For assistance, contact Support. • Conflicting values provided in HTTP headers and query par ameters. • Conflicting values provided in HTTP headers and POST form fields. • CopyObject request made on objects larger than 5GB in size.	

HTTP status code	Error code	Description of error code	
	InvalidSessionException	Returned if the session doesn't exist anymore because it timed out or expired.	
	InvalidSignature	The request signature that the server calculated does not match the signature that you provided. Check your AWS secret access key and signing method. For more information, see Signing and authenticating REST requests.	
	InvalidSOAPRequest	The SOAP request body is not valid.	
	InvalidStorageClass	The storage class that you specified is not valid.	
	InvalidTag	Your request contains tag input that is not valid. For example, your request might contain duplicate keys, keys or values that are too long, or system tags.	
	InvalidToken	The provided token is malformed or otherwise not valid.	

HTTP status code	Error code	Description of error code	
	InvalidURI	The specified URI couldn't be parsed.	
	KeyTooLongError	Your key is too long.	
	KMS.DisabledException	The request was rejected because the specified KMS key is not enabled.	

HTTP status code	Error code	Description of error code	
	KMS.InvalidKeyUsageException	The request was rejected for one of the following reasons: • The KeyUsage value of the KMS key is incompatible with the API operation. • The encryption algorithm or signing algorithm specified for the operation is incompatible with the type of key material in the KMS key (KeySpec). For encrypting, decrypting, re-encrypting, and generating data keys, the KeyUsage must be ENCRYPT_DECRYPT. For signing and verifying messages, the KeyUsage must be SIGN_VERIFY. For generating and verifying message authentication codes (MACs), the KeyUsage must be GENERATE_VERIFY_MAC. For deriving key agreement secrets, the KeyUsage	

HTTP status code	Error code	Description of error code	
		must be KEY_AGREEMENT. To find the KeyUsage of a KMS key, use the DescribeKey operation. To find the encryption or signing algorithms supported for a particular KMS key, use the DescribeKey operation.	

HTTP status code	Error code	Description of error code	
	KMS.KMSInvalidStateException	The request was rejected because the state of the specified resource is not valid for this request. This exception means one of the following: • The key state of the KMS key is not compatible with the operation. To find the key state, use the DescribeKey operation. For more information about which key states are compatible with each KMS operation, see Key states of AWS KMS keys in the *AWS Key Management Service Developer Guide*. • For cryptographic operations on KMS keys in custom key stores, this exception represents a general failure with many possible causes. To identify the cause, see the error message	

HTTP status code	Error code	Description of error code	
		that accompanies the exception.	
	KMS.NotFoundException	The request was rejected because the specified entity or resource could not be found.	
	LambdaInvalidResponse	Returned to the original caller when WriteGetObjectResponse responds with ValidationError to AWS Lambda. See the ValidationError message for more details. Not all cases of ValidationError result in a LambdaInvalidResponse error.	

HTTP status code	Error code	Description of error code	
	LambdaInvocationFailed	Lambda function invocation failed. Caller s might receive the following error when S3 Object Lambda is unable to successfully invoke the configure d Lambda function. The error message might contain details about an eventual error returned by the AWS Lambda service when invoking the function (for example, status code, error code, error message and request ID).	
	MalformedACLError	The ACL that you provided was not well formed or did not validate against our published schema.	
	MalformedPOSTRequest	The body of your POST request is not well-form ed multipart/form-data.	
	MalformedXML	The XML that you provided was not well formed or did not validate against our published schema.	

HTTP status code	Error code	Description of error code	
	MaxPostPreDataLengthExceededError	Your POST request fields preceding the upload file were too large.	
	MetadataTooLarge	Your metadata headers exceed the maximum allowed metadata size.	
	MissingAttachment	A SOAP attachment was expected, but none was found.	
	MissingRequestBodyError	You sent an empty XML document as a request.	
	MissingSecurityHeader	Your request is missing a required header.	
	NoLoggingStatusForKey	There is no such thing as a logging status subresource for a key.	
	NotDeviceOwnerError	The device that generated the token is not owned by the authenticated user.	
	ResponseInterrupted	Returned to the original caller when an error is encountered while reading the WriteGetObjectResponse body.	

HTTP status code	Error code	Description of error code	
	RequestHeaderSectionTooLarge	The request header and query parameters used to make the request exceed the maximum allowed sizes	
	TokenCodeInvalidError	The serial number and/or token code you provided is not valid.	
	UnexpectedContent	This request contains unsupported content.	
	UnsupportedArgument	The request contained an unsupported argument.	
	UnsupportedSignature	The provided request is signed with an unsupported STS Token version or the signature version is not supported.	
	UserKeyMustBeSpecified	The bucket POST request must contain the specified field name. If it is specified, check the order of the fields.	
	IncorrectEndpoint	The specified bucket exists in another Region. Direct requests to the correct endpoint.	

HTTP status code	Error code	Description of error code
	ValidationError	Validation errors might be returned from the WriteGetObjectResponse API operation and can occur for numerous reasons. See the error message for more details.
403 Forbidden	RequestTimeTooSkewed	The difference between the request time and the server's time is too large.
	SignatureDoesNotMatch	The request signature that the server calculated does not match the signature that you provided. Check your AWS secret access key and signing method. For more information, see REST Authentication and SOAP Authentication.
	NotSignedUp	Your account is not signed up for the Amazon S3 service. You must sign up before you can use Amazon S3. You can sign up at the following URL: https://aws.amazon.com/s3

HTTP status code	Error code	Description of error code
	InvalidSecurity	The provided security credentials are not valid.
	InvalidPayer	All access to this object has been disabled. For further assistance, see Contact Us.
	InvalidAccessKeyId	The AWS access key ID that you provided does not exist in our records.
	AccountProblem	There is a problem with your AWS account that prevents the operation from completing s uccessfully. For further assistance, see Contact Us.
	UnauthorizedAccessError	Applicable in China Regions only. Returned when a request is made to a bucket that doesn't have an ICP license. For more information, see ICP Recordal.
	UnexpectedIPError	Applicable in China Regions only. This request was rejected because the IP was unexpected.

HTTP status code	Error code	Description of error code
	MissingAuthenticationToken	The request was not signed.
	LambdaPermissionError	The caller is not authorized to invoke the Lambda function. The caller must have permission to invoke the Lambda function. Check the policies attached to the caller and ensure that they've been allowed to use `lambda:Invoke` for the configured function. The error message might contain details about an eventual error returned by the Lambda service when invoking the function (for example, status code, error code, error message and request ID).

HTTP status code	Error code	Description of error code	
404 Not Found	LambdaNotFound	The AWS Lambda function was not found. The configured Lambda function, version, or alias was not found when attempting to invoke it. Ensure that the S3 Object Lambda Access Point configuration points to the correct Lambda function ARN. The error message might contain details about an eventual error returned by the AWS Lambda service when invoking the function (for example, status code, error code, error message and request ID).	
	NoSuchAsyncRequest	The specified request was not found.	
	NoSuchObjectLockConfiguration	The specified object does not have an ObjectLock configuration.	

HTTP status code	Error code	Description of error code
	NoSuchUpload	The specified multipart upload does not exist. The upload ID might not be valid, or the multipart upload might have been aborted or completed.
	NoSuchWebsiteConfiguration	The specified bucket does not have a website configuration.
	NoTransformationDefined	No transformation found for this Object Lambda Access Point.
	ObjectLockConfigurationNotFoundError	The Object Lock configuration does not exist for this bucket.
405 Method Not Allowed	MethodNotAllowed	The specified method is not allowed against this resource.
409 Conflict	BucketAlreadyExists	The requested bucket name is not available. The bucket namespace is shared by all users of the system. Specify a different name and try again.

HTTP status code	Error code	Description of error code
	InvalidBucketState	The request is not valid for the current state of the bucket.
	OperationAborted	A conflicting conditional operation is currently in progress against this resource. Try again.
411 Length Required	MissingContentLength	You must provide the Content-Length HTTP header.
412 Precondition Failed	RequestIsNotMultiPartContent	A bucket POST request must be of the enclosure-type multipart/form-data.
416 Requested Range Not Satisfiable	InvalidRange	The requested range is not valid for the request. Try another range.

Understanding and managing Amazon S3 storage classes

Each object in Amazon S3 has a storage class associated with it. By default, objects in S3 are stored in the S3 Standard storage class, however Amazon S3 offers a range of other storage classes for the objects that you store. You choose a class depending on your use case scenario and performance access requirements. Choosing a storage class designed for your use case lets you optimize storage costs, performance, and availability for your objects. All of these storage classes offer high durability.

The following sections provide details of the various storage classes and how to set the storage class for your objects.

Topics

- Storage classes for frequently accessed objects
- Storage class for automatically optimizing data with changing or unknown access patterns
- Storage classes for infrequently accessed objects
- Storage classes for rarely accessed objects
- Storage class for Amazon S3 on Outposts
- Comparing the Amazon S3 storage classes
- Setting the storage class of an object
- Amazon S3 analytics – Storage Class Analysis
- Managing storage costs with Amazon S3 Intelligent-Tiering
- Understanding S3 Glacier storage classes for long-term data storage
- Working with archived objects

Storage classes for frequently accessed objects

For performance-sensitive use cases (those that require millisecond access time) and frequently accessed data, Amazon S3 provides the following storage classes:

- **S3 Standard** (`STANDARD`) – The default storage class. If you don't specify the storage class when you upload an object, Amazon S3 assigns the S3 Standard storage class. To help you optimize costs between S3 Standard and S3 Standard-IA you can use Amazon S3 analytics – Storage Class Analysis.
- **S3 Express One Zone** (`EXPRESS_ONEZONE`) – Amazon S3 Express One Zone is a high-performance, single-zone Amazon S3 storage class that is purpose-built to deliver consistent, single-digit millisecond data access for your most latency-sensitive applications. S3 Express One Zone is the lowest latency cloud object storage class available today, with data access speed up to 10x faster and with request costs 50 percent lower than S3 Standard. With S3 Express One Zone, your data is redundantly stored on multiple devices within a single Availability Zone. For more information, see S3 Express One Zone.

- **Reduced Redundancy Storage** (REDUCED_REDUNDANCY) – The Reduced Redundancy Storage (RRS) class is designed for noncritical, reproducible data that can be stored with less redundancy than the S3 Standard storage class.

  > **Important**
  >
  > We recommend not using this storage class. The S3 Standard storage class is more cost-effective.

  For durability, RRS objects have an average annual expected loss of 0.01 percent of objects. If an RRS object is lost, when requests are made to that object, Amazon S3 returns a 405 error.

Storage class for automatically optimizing data with changing or unknown access patterns

S3 Intelligent-Tiering (INTELLIGENT_TIERING) is an Amazon S3 storage class that's designed to optimize storage costs by automatically moving data to the most cost-effective access tier, without performance impact or operational overhead. S3 Intelligent-Tiering is the only cloud storage class that delivers automatic cost savings by moving data on a granular object level between access tiers when access patterns change. S3 Intelligent-Tiering is the ideal storage class when you want to optimize storage costs for data that has unknown or changing access patterns. There are no retrieval fees for S3 Intelligent-Tiering.

For a small monthly object monitoring and automation fee, S3 Intelligent-Tiering monitors access patterns and automatically moves objects that have not been accessed to lower-cost access tiers. S3 Intelligent-Tiering delivers automatic storage cost savings in three low-latency and high-throughput access tiers. For data that can be accessed asynchronously, you can choose to activate automatic archiving capabilities within the S3 Intelligent-Tiering storage class. S3 Intelligent-Tiering is designed for 99.9% availability and 99.999999999% durability.

S3 Intelligent-Tiering automatically stores objects in three access tiers:

- **Frequent Access** – Objects that are uploaded or transitioned to S3 Intelligent-Tiering are automatically stored in the Frequent Access tier.
- **Infrequent Access** – S3 Intelligent-Tiering moves objects that have not been accessed in 30 consecutive days to the Infrequent Access tier.

- **Archive Instant Access** – With S3 Intelligent-Tiering, any existing objects that have not been accessed for 90 consecutive days are automatically moved to the Archive Instant Access tier.

In addition to these three tiers, S3 Intelligent-Tiering offers two optional archive access tiers:

- **Archive Access** – S3 Intelligent-Tiering provides you with the option to activate the Archive Access tier for data that can be accessed asynchronously. After activation, the Archive Access tier automatically archives objects that have not been accessed for a minimum of 90 consecutive days.
- **Deep Archive Access** – S3 Intelligent-Tiering provides you with the option to activate the Deep Archive Access tier for data that can be accessed asynchronously. After activation, the Deep Archive Access tier automatically archives objects that have not been accessed for a minimum of 180 consecutive days.

Note

- Only activate the Archive Access tier for 90 days if you want to bypass the Archive Instant Access tier. The Archive Access tier delivers slightly lower-cost storage with minute-to-hour retrieval times. The Archive Instant Access tier delivers millisecond access and high-throughput performance.
- Activate the Archive Access and Deep Archive Access tiers only if your objects can be accessed asynchronously by your application. If the object that you are retrieving is stored in the Archive Access or Deep Archive Access tiers, first restore the object by using `RestoreObject`.

You can move newly created data to S3 Intelligent-Tiering, setting it as your default storage class. You can also choose to activate one or both of the archive access tiers by using the `PutBucketIntelligentTieringConfiguration` API operation, the AWS CLI, or the Amazon S3 console. For more information about using S3 Intelligent-Tiering and activating the archive access tiers, see Using S3 Intelligent-Tiering.

To access objects in the Archive Access or Deep Archive Access tiers, you first need to restore them. For more information, see Restoring objects from the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers.

Note

If the size of an object is less than 128 KB, it is not monitored and not eligible for auto-tiering. Smaller objects are always stored in the Frequent Access tier. For more information about S3 Intelligent-Tiering, see S3 Intelligent-Tiering access tiers.

Storage classes for infrequently accessed objects

The **S3 Standard-IA** and **S3 One Zone-IA** storage classes are designed for long-lived and infrequently accessed data. (IA stands for *infrequent access*.) S3 Standard-IA and S3 One Zone-IA objects are available for millisecond access (similar to the S3 Standard storage class). Amazon S3 charges a retrieval fee for these objects, so they are most suitable for infrequently accessed data. For pricing information, see Amazon S3 pricing.

For example, you might choose the S3 Standard-IA and S3 One Zone-IA storage classes to do the following:

- For storing backups.
- For older data that is accessed infrequently, but that still requires millisecond access. For example, when you upload data, you might choose the S3 Standard storage class, and use lifecycle configuration to tell Amazon S3 to transition the objects to the S3 Standard-IA or S3 One Zone-IA class.

  For more information about lifecycle management, see Managing the lifecycle of objects.

Note

The S3 Standard-IA and S3 One Zone-IA storage classes are suitable for objects larger than 128 KB that you plan to store for at least 30 days. If an object is less than 128 KB, Amazon S3 charges you for 128 KB. If you delete an object before the end of the 30-day minimum storage duration period, you are charged for 30 days. Objects that are deleted, overwritten, or transitioned to a different storage class before 30 days will incur the normal storage usage charge plus a pro-rated charge for the remainder of the 30-day minimum. For pricing information, see Amazon S3 pricing.

These storage classes differ as follows:

- **S3 Standard-IA** (`STANDARD_IA`) – Amazon S3 stores the object data redundantly across multiple geographically separated Availability Zones (similar to the S3 Standard storage class). S3 Standard-IA objects are resilient to the loss of an Availability Zone. This storage class offers greater availability and resiliency than the S3 One Zone-IA class. To help you optimize costs between S3 Standard and S3 Standard-IA you can use Amazon S3 analytics – Storage Class Analysis
- **S3 One Zone-IA** (`ONEZONE_IA`) – Amazon S3 stores the object data in only one Availability Zone, which makes it less expensive than S3 Standard-IA. However, the data is not resilient to the physical loss of the Availability Zone resulting from disasters, such as earthquakes and floods. The S3 One Zone-IA storage class is as durable as S3 Standard-IA, but it is less available and less resilient. For a comparison of storage class durability and availability, see Comparing the Amazon S3 storage classes at the end of this section. For pricing information, see Amazon S3 pricing. For data residency and isolation use cases, you can create directory buckets in AWS Local Zones and use the S3 Express One Zone (`EXPRESS_ONEZONE`) and S3 One Zone-IA (`ONEZONE_IA`) storage classes. For more information about directory buckets in Local Zones, see Data residency workloads.

We recommend the following:

- **S3 Standard-IA** (`STANDARD_IA`) – Use for your primary or only copy of data that can't be re-created.
- **S3 One Zone-IA** (`ONEZONE_IA`) – Use if you can re-create the data if the Availability Zone fails, for object replicas when configuring S3 Cross-Region Replication (CRR). Also, for data residency and isolation, you can create directory buckets in AWS Local Zones and use the S3 One Zone-IA storage class.

Storage classes for rarely accessed objects

The **S3 Glacier Instant Retrieval** (`GLACIER_IR`), **S3 Glacier Flexible Retrieval** (`GLACIER`), and **S3 Glacier Deep Archive** (`DEEP_ARCHIVE`) storage classes are designed for low-cost, long-term data storage and data archiving. These storage classes require minimum storage durations and retrieval fees making them most effective for rarely accessed data. For more information about S3 Glacier storage classes, see Understanding S3 Glacier storage classes for long-term data storage.

Amazon S3 provides the following S3 Glacier storage classes:

- **S3 Glacier Instant Retrieval** (GLACIER_IR) – Use for long-term data that's rarely accessed and requires milliseconds retrieval. Data in this storage class is available for real-time access.
- **S3 Glacier Flexible Retrieval** (GLACIER) – Use for archives where portions of the data might need to be retrieved in minutes. Data in this storage class is archived, and not available for real-time access.
- **S3 Glacier Deep Archive** (DEEP_ARCHIVE) – Use for archiving data that rarely needs to be accessed. Data in this storage class is archived, and not available for real-time access.

Retrieving archived objects

You can set the storage class of an object to S3 Glacier Flexible Retrieval (GLACIER) or S3 Glacier Deep Archive (DEEP_ARCHIVE) in the same ways that you do for the other storage classes as described in the section Setting the storage class of an object. However, S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive objects are archived, and not available for real-time access. For more information, see Understanding archival storage in S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive.

> **Note**
>
> When you use S3 Glacier storage classes, your objects remain in Amazon S3. You can't access them directly through the separate Amazon Glacier service. For information about the Amazon Glacier service, see the Amazon Glacier Developer Guide.

Storage class for Amazon S3 on Outposts

With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts resources and store and retrieve objects on-premises for applications that require local data access, local data processing, and data residency. You can use the same API operations and features on AWS Outposts as you do on Amazon S3, including access policies, encryption, and tagging. You can use S3 on Outposts through the AWS Management Console, AWS CLI, AWS SDKs, or REST API.

S3 on Outposts provides a new storage class, S3 Outposts (OUTPOSTS). The S3 Outposts storage class is available only for objects stored in buckets on Outposts. If you try to use this storage class with an S3 bucket in an AWS Region, an `InvalidStorageClass` error occurs. In addition, if you try to use other S3 storage classes with objects stored in S3 on Outposts buckets, the same error occurs.

Objects stored in the S3 Outposts (0UTP0STS) storage class are always encrypted by using server-side encryption with Amazon S3 managed encryption keys (SSE-S3). For more information, see Using server-side encryption with Amazon S3 managed keys (SSE-S3).

You can also explicitly choose to encrypt objects stored in the S3 Outposts storage class by using server-side encryption with customer-provided encryption keys (SSE-C). For more information, see Using server-side encryption with customer-provided keys (SSE-C).

> **Note**
>
> S3 on Outposts doesn't support server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS).

For more information about S3 on Outposts, see What is S3 on Outposts in the *Amazon S3 on Outposts User Guide*.

Comparing the Amazon S3 storage classes

The following table compares the storage classes, including their availability, durability, minimum storage duration, and other considerations.

Storage Class	Designed for	Durability (designed for)	Availability (designed for)	Availability Zones	Min storage duration	Min billable object size	Other Considerations
STANDARD	Frequently accessed data	99.999999999%	99.99%	>= 3	None	None	None
STANDARD_IA	Long-lived, infrequently accessed data	99.999999999%	99.9%	>= 3	30 days	128 KB	Per GB retrieval fees apply.
INTELLIGENT_TIERING	Long-lived data with changing or unknown access patterns	99.999999999%	99.9%	>= 3	30 days	None	Monitoring and automation fees per object apply. No retrieval fees.
ONEZONE_IA	Long-lived, infrequently accessed, non-critical data	99.999999999%	99.5%	1	30 days	128 KB	Per GB retrieval fees apply. Not resilient to the loss of the Availability Zone.
GLACIER	Long-term data archiving with retrieval times ranging from minutes to hours	99.999999999%	99.99% (after you restore objects)	>= 3	90 days	None	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.
DEEP_ARCHIVE	Archiving rarely accessed data with a default retrieval time of 12 hours	99.999999999%	99.99% (after you restore objects)	>= 3	180 days	None	Per GB retrieval fees apply. You must first restore archived objects before you can access them. For more information, see Restoring Archived Objects.
RRS (Not recommended)	Frequently accessed, non-critical data	99.99%	99.99%	>= 3	None	None	None

* S3 Glacier Flexible Retrieval requires 40 KB of additional metadata for each archived object. This includes 32 KB of metadata charged at the S3 Glacier Flexible Retrieval rate (required to identify

and retrieve your data), and an additional 8 KB data charged at the S3 Standard rate. The S3 Standard rate is required to maintain the user-defined name and metadata for objects archived to S3 Glacier Flexible Retrieval. For more information about storage classes, see Amazon S3 storage classes.

** S3 Glacier Deep Archive requires 40 KB of additional metadata for each archived object. This includes 32 KB of metadata charged at the S3 Glacier Deep Archive rate (required to identify and retrieve your data), and an additional 8 KB data charged at the S3 Standard rate. The S3 Standard rate is required to maintain the user-defined name and metadata for objects archived to Amazon S3 Glacier Deep Archive. For more information about storage classes, see Amazon S3 storage classes.

Be aware that all of the storage classes except for S3 One Zone-IA (`ONEZONE_IA`) and S3 Express One Zone (`EXPRESS_ONEZONE`) are designed to be resilient to the physical loss of an Availability Zone resulting from disasters. Also, consider costs, in addition to the performance requirements of your application scenario. For storage class pricing, see Amazon S3 pricing.

Setting the storage class of an object

You can specify a storage class for an object when you upload it. If you don't, Amazon S3 uses the default Amazon S3 Standard storage class for objects in general purpose buckets. You can also change the storage class of an object that's already stored in an Amazon S3 general purpose bucket to any other storage class using the Amazon S3 console, AWS SDKs, or the AWS Command Line Interface (AWS CLI). All of these approaches use Amazon S3 API operations to send requests to Amazon S3.

> **Note**
>
> You can't change the storage class of objects stored in directory buckets.

You can direct Amazon S3 to change the storage class of objects automatically by adding an S3 Lifecycle configuration to a bucket. For more information, see Managing the lifecycle of objects.

When setting up a S3 Replication configuration, you can set the storage class for replicated objects to any other storage class. However, you can't replicate objects that are stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. For more information, see Replication configuration file elements.

When setting the storage class programmatically you provide the value of the storage class. The following is a list of console names for storage classes with their corresponding API values:

- **Reduced Redundancy Storage** – `REDUCED_REDUNDANCY`
- **S3 Express One Zone** – `EXPRESS_ONEZONE`
- **S3 Glacier Deep Archive** – `DEEP_ARCHIVE`
- **S3 Glacier Flexible Retrieval** – `GLACIER`
- **S3 Glacier Instant Retrieval** – `GLACIER_IR`
- **S3 Intelligent-Tiering** – `INTELLIGENT_TIERING`
- **S3 One Zone-IA** – `ONEZONE_IA`
- **S3 Standard** – `STANDARD`
- **S3 Standard-IA** – `STANDARD_IA`

Setting the storage class on a new object

To set the storage class when you upload an object, you can use the following methods.

Using the S3 console

To set the storage class when uploading a new object in the console:

1. Sign in to the AWS Management Console and open the Amazon S3 console at: https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to upload your folders or files to.
4. Choose **Upload**.
5. In the **Upload** window, choose **Properties**.
6. Under Storage class, choose a storage classes for the files you're uploading.
7. (Optional) Configure any additional properties for the files you're uploading, For more information, see Uploading objects
8. In the Upload window, do one of the following:
    - Drag files and folders to the Upload window.

- Choose **Add file** or **Add folder**, choose the files or folders to upload, and choose **Open**.

9. Choose **Upload**.

Using the REST API

You can specify the storage class on an object when you create it using the `PutObject`, `POST Object` Object, and `CreateMultipartUpload` API operations, add the `x-amz-storage-class` request header. If you don't add this header, Amazon S3 uses the default S3 Standard (`STANDARD`) storage class.

This example request uses the `PutObject` command to set the storage class on a new object to S3 Intelligent-Tiering:

```
PUT /my-image.jpg HTTP/1.1
Host: amzn-s3-demo-bucket1.s3.Region.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
Content-Type: image/jpeg
Content-Length: 11434
Expect: 100-continue
x-amz-storage-class: INTELLIGENT_TIERING
```

Using the AWS CLI

This example uses the `put-object` command to upload the *my_images.tar.bz2* to *amzn-s3-demo-bucket1* in the *GLACIER* storage class:

```
aws s3api put-object --bucket amzn-s3-demo-bucket1 --key dir-1/my_images.tar.bz2 --
storage-class GLACIER --body my_images.tar.bz2
```

If the object size is more than 5 GB, use the following command to set the storage class:

```
aws s3 cp large_test_file s3://amzn-s3-demo-bucket1 --storage-class GLACIER
```

Changing the storage class for an existing object

To set the storage class when you upload an object, you can use the following methods.

Using the S3 console

You can change an object's storage class using the Amazon S3 console if the object size is less than 5 GB. If larger, we recommend adding an S3 Lifecycle configuration to change the object's storage class.

To change the storage class of an object in the console:

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket containing the objects you want to change.
4. Select the check box to the left of the names of the objects you want to change.
5. On the **Actions** menu, choose **Edit storage class** from the list of options that appears.
6. Select from the storage classes available for your object.
7. Under **Additional copy settings**, choose whether you want to **Copy source settings**, **Don't specify settings**, or **Specify settings**. **Copy source settings** is the default option. If you only want to copy the object without the source settings attributes, choose **Don't specify settings**. Choose **Specify settings** to specify settings for storage class, ACLs, object tags, metadata, server-side encryption, and additional checksums.
8. Choose **Save changes**. Amazon S3 saves your changes.

Using the REST API

To change the storage class of an existing object, use the following methods.

To change the storage class of an existing object, you make a copy of the object to the same location with a new storage class using the `CopyObject` operation. The following example request copies an object to itself with the storage class changed to S3 Intelligent-Tiering:

```
PUT /my-image.jpg HTTP/1.1
```

```
Host: amzn-s3-demo-bucket1.s3.Region.amazonaws.com
Date: Wed, 12 Oct 2009 17:50:00 GMT
Authorization: authorization string
x-amz-copy-source: /amzn-s3-demo-bucket1/my-image.jpg
x-amz-metadata-directive: COPY
x-amz-storage-class: INTELLIGENT_TIERING
```

Using the AWS CLI

This example uses the cp command to change the storage class of the of an existing object from its current storage class to *DEEP_ARCHIVE* storage class:

```
aws s3 cp object_S3_URI object_S3_URI --storage-class DEEP_ARCHIVE
```

Restricting access policy permissions to a specific storage class

When you grant access policy permissions for Amazon S3 operations, you can use the `s3:x-amz-storage-class` condition key to restrict which storage class to use when storing uploaded objects. For example, when you grant the `s3:PutObject` permission, you can restrict object uploads to a specific storage class. For an example policy, see Example: Restricting object uploads to objects with a specific storage class.

For more information about using conditions in policies and a complete list of Amazon S3 condition keys, see the following topics:

- Actions, resources, and condition keys for Amazon S3 in the *Service Authorization Reference*

  For more information about the permissions to S3 API operations by S3 resource types, see Required permissions for Amazon S3 API operations.
- Bucket policy examples using condition keys

Amazon S3 analytics – Storage Class Analysis

By using Amazon S3 analytics *Storage Class Analysis* you can analyze storage access patterns to help you decide when to transition the right data to the right storage class. This new Amazon S3 analytics feature observes data access patterns to help you determine when to transition less

frequently accessed STANDARD storage to the STANDARD_IA (IA, for infrequent access) storage class. For more information about storage classes, see Understanding and managing Amazon S3 storage classes.

After storage class analysis observes the infrequent access patterns of a filtered set of data over a period of time, you can use the analysis results to help you improve your lifecycle configurations. You can configure storage class analysis to analyze all the objects in a bucket. Or, you can configure filters to group objects together for analysis by common prefix (that is, objects that have names that begin with a common string), by object tags, or by both prefix and tags. You'll most likely find that filtering by object groups is the best way to benefit from storage class analysis.

> **Important**
>
> Storage class analysis only provides recommendations for Standard to Standard IA classes.

You can have multiple storage class analysis filters per bucket, up to 1,000, and will receive a separate analysis for each filter. Multiple filter configurations allow you analyze specific groups of objects to improve your lifecycle configurations that transition objects to STANDARD_IA.

Storage class analysis provides storage usage visualizations in the Amazon S3 console that are updated daily. You can also export this daily usage data to an S3 bucket and view them in a spreadsheet application, or with business intelligence tools, like Quick.

There are costs associated with the storage class analysis. For pricing information, see *Management and insights* Amazon S3 pricing.

Topics

- How do I set up storage class analysis?
- How do I use storage class analysis?
- How can I export storage class analysis data?
- Configuring storage class analysis

How do I set up storage class analysis?

You set up storage class analysis by configuring what object data you want to analyze. You can configure storage class analysis to do the following:

- **Analyze the entire contents of a bucket.**

  You'll receive an analysis for all the objects in the bucket.

- **Analyze objects grouped together by prefix and tags.**

  You can configure filters that group objects together for analysis by prefix, or by object tags, or by a combination of prefix and tags. You receive a separate analysis for each filter you configure. You can have multiple filter configurations per bucket, up to 1,000.

- **Export analysis data.**

  When you configure storage class analysis for a bucket or filter, you can choose to have the analysis data exported to a file each day. The analysis for the day is added to the file to form a historic analysis log for the configured filter. The file is updated daily at the destination of your choice. When selecting data to export, you specify a destination bucket and optional destination prefix where the file is written.

You can use the Amazon S3 console, the REST API, or the AWS CLI or AWS SDKs to configure storage class analysis.

- For information about how to configure storage class analysis in the Amazon S3 console, see Configuring storage class analysis.
- To use the Amazon S3 API, use the PutBucketAnalyticsConfiguration REST API, or the equivalent, from the AWS CLI or AWS SDKs.

How do I use storage class analysis?

You use storage class analysis to observe your data access patterns over time to gather information to help you improve the lifecycle management of your STANDARD_IA storage. After you configure a filter, you'll start seeing data analysis based on the filter in the Amazon S3 console in 24 to 48 hours. However, storage class analysis observes the access patterns of a filtered data set for 30 days or longer to gather information for analysis before giving a result. The analysis continues to run after the initial result and updates the result as the access patterns change

When you first configure a filter, the Amazon S3 console may take a moment to analyze your data.

Storage class analysis observes the access patterns of a filtered object data set for 30 days or longer to gather enough information for the analysis. After storage class analysis has gathered sufficient information, you'll see a message in the Amazon S3 console that analysis is complete.

When performing the analysis for infrequently accessed objects storage class analysis looks at the filtered set of objects grouped together based on age since they were uploaded to Amazon S3. Storage class analysis determines if the age group is infrequently accessed by looking at the following factors for the filtered data set:

- Objects in the STANDARD storage class that are larger than 128 KB.
- How much average total storage you have per age group.
- Average number of bytes transferred out (not frequency) per age group.
- Analytics export data only includes requests with data relevant to storage class analysis. This might cause differences in the number of requests, and the total upload and request bytes compared to what are shown in storage metrics or tracked by your own internal systems.
- Failed GET and PUT requests are not counted for the analysis. However, you will see failed requests in storage metrics.

How Much of My Storage did I Retrieve?

The Amazon S3 console graphs how much of the storage in the filtered data set has been retrieved for the observation period.

What Percentage of My Storage did I Retrieve?

The Amazon S3 console also graphs what percentage of the storage in the filtered data set has been retrieved for the observation period.

As stated earlier in this topic, when you are performing the analysis for infrequently accessed objects, storage class analysis looks at the filtered set of objects grouped together based on the age since they were uploaded to Amazon S3. The storage class analysis uses the following predefined object age groups:

- Amazon S3 Objects less than 15 days old
- Amazon S3 Objects 15-29 days old
- Amazon S3 Objects 30-44 days old
- Amazon S3 Objects 45-59 days old
- Amazon S3 Objects 60-74 days old
- Amazon S3 Objects 75-89 days old
- Amazon S3 Objects 90-119 days old

- Amazon S3 Objects 120-149 days old
- Amazon S3 Objects 150-179 days old
- Amazon S3 Objects 180-364 days old
- Amazon S3 Objects 365-729 days old
- Amazon S3 Objects 730 days and older

Usually it takes about 30 days of observing access patterns to gather enough information for an analysis result. It might take longer than 30 days, depending on the unique access pattern of your data. However, after you configure a filter you'll start seeing data analysis based on the filter in the Amazon S3 console in 24 to 48 hours. You can see analysis on a daily basis of object access broken down by object age group in the Amazon S3 console.

How Much of My Storage is Infrequently Accessed?

The Amazon S3 console shows the access patterns grouped by the predefined object age groups. The **Frequently accessed** or **Infrequently accessed** text shown is meant as a visual aid to help you in the lifecycle creation process.

How can I export storage class analysis data?

You can choose to have storage class analysis export analysis reports to a comma-separated values (CSV) flat file. Reports are updated daily and are based on the object age group filters you configure. When using the Amazon S3 console you can choose the export report option when you create a filter. When selecting data export you specify a destination bucket and optional destination prefix where the file is written. You can export the data to a destination bucket in a different account. The destination bucket must be in the same region as the bucket that you configure to be analyzed.

You must create a bucket policy on the destination bucket to grant permissions to Amazon S3 to verify what AWS account owns the bucket and to write objects to the bucket in the defined location. For an example policy, see Grant permissions for S3 Inventory and S3 analytics.

After you configure storage class analysis reports, you start getting the exported report daily after 24 hours. After that, Amazon S3 continues monitoring and providing daily exports.

You can open the CSV file in a spreadsheet application or import the file into other applications like AWS Quick. For information on using Amazon S3 objects with AWS Quick, see the AWS Quick User Guide.

Data in the exported file is sorted by date within object age group as shown in following examples. If the storage class is STANDARD the row also contains data for the columns `ObjectAgeForSIATransition` and `RecommendedObjectAgeForSIATransition`.

Date	ConfigId	Filter	StorageClass	ObjectAge	ObjectCount	DataUploaded_MB	Storage_MB	DataRetrieved_MB	GetRequestCount	CumulativeAccessRatio	ObjectAgeForSIATransition	RecommendedObjectAgeForSIATransition
8/17/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/2/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/22/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
8/25/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/6/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/30/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/28/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		
8/21/2021	SalesMaterial	SalesMaterial	STANDARD	000-014			0.4313			0		
9/5/2021	SalesMaterial	SalesMaterial	STANDARD	000-014						0.04096734		

At the end of the report the object age group is given as ALL. The ALL rows contain cumulative totals, including objects smaller than 128 KB, for all the age groups for that day.

8/24/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014
9/3/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0.02426125	015-029
8/28/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0.03545875	015-029
8/17/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014
8/25/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014
9/6/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0.0209529	015-029
9/4/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0.02304819	015-029
8/22/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014
8/21/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014
8/30/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0.03073092	015-029
8/20/2021	SalesMaterial	SalesMaterial	STANDARD	ALL	3		0.4599			0	000-014

The next section describes the columns used in the report.

Exported file layout

The following table describes the Amazon S3 storage class analysis export file layout.

Configuring storage class analysis

By using the Amazon S3 analytics storage class analysis tool, you can analyze storage access patterns to help you decide when to transition the right data to the right storage class. Storage class analysis observes data access patterns to help you determine when to transition less frequently accessed STANDARD storage to the STANDARD_IA (IA, for infrequent access) storage class. For more information about STANDARD_IA, see the Amazon S3 FAQ and Understanding and managing Amazon S3 storage classes.

You set up storage class analysis by configuring what object data you want to analyze. You can configure storage class analysis to do the following:

- **Analyze the entire contents of a bucket.**

  You'll receive an analysis for all the objects in the bucket.

- **Analyze objects grouped together by prefix and tags.**

You can configure filters that group objects together for analysis by prefix, or by object tags, or by a combination of prefix and tags. You receive a separate analysis for each filter you configure. You can have multiple filter configurations per bucket, up to 1,000.

- **Export analysis data.**

  When you configure storage class analysis for a bucket or filter, you can choose to have the analysis data exported to a file each day. The analysis for the day is added to the file to form a historic analysis log for the configured filter. The file is updated daily at the destination of your choice. When selecting data to export, you specify a destination bucket and optional destination prefix where the file is written.

You can use the Amazon S3 console, the REST API, or the AWS CLI or AWS SDKs to configure storage class analysis.

> **Important**
>
> Storage class analysis does not give recommendations for transitions to the ONEZONE_IA or S3 Glacier Flexible Retrieval storage classes.
> If you want to configure storage class analysis to export your findings as a .csv file and the destination bucket uses default bucket encryption with a AWS KMS key, you must update the AWS KMS key policy to grant Amazon S3 permission to encrypt the .csv file. For instructions, see Granting Amazon S3 permission to use your customer managed key for encryption.

For more information about analytics, see Amazon S3 analytics – Storage Class Analysis.

Using the S3 console

To configure storage class analysis

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets** or **Directory buckets**.
3. In the buckets list, choose the name of the bucket for which you want to configure storage class analysis.
4. Choose the **Metrics** tab.

5. Under **Storage Class Analysis**, choose **Create analytics configuration**.
6. Type a name for the filter. If you want to analyze the whole bucket, leave the **Prefix** field empty.
7. In the **Prefix** field, type text for the prefix for the objects that you want to analyze.
8. To add a tag, choose **Add tag**. Enter a key and value for the tag. You can enter one prefix and multiple tags.
9. Optionally, you can choose **Enable** under **Export CSV** to export analysis reports to a comma-separated values (.csv) flat file. Choose a destination bucket where the file can be stored. You can type a prefix for the destination bucket. The destination bucket must be in the same AWS Region as the bucket for which you are setting up the analysis. The destination bucket can be in a different AWS account.

   If the destination bucket for the .csv file uses default bucket encryption with a KMS key, you must update the AWS KMS key policy to grant Amazon S3 permission to encrypt the .csv file. For instructions, see Granting Amazon S3 permission to use your customer managed key for encryption.
10. Choose **Create Configuration**.

Amazon S3 creates a bucket policy on the destination bucket that grants Amazon S3 write permission. This will allow it to write the export data to the bucket.

If an error occurs when you try to create the bucket policy, you'll be given instructions on how to fix it. For example, if you chose a destination bucket in another AWS account and do not have permissions to read and write to the bucket policy, you'll see the following message. You must have the destination bucket owner add the displayed bucket policy to the destination bucket. If the policy is not added to the destination bucket you won't get the export data because Amazon S3 doesn't have permission to write to the destination bucket. If the source bucket is owned by a different account than that of the current user, then the correct account ID of the source bucket must be substituted in the policy.

For information about the exported data and how the filter works, see Amazon S3 analytics – Storage Class Analysis.

Using the REST API

To configure Storage Class Analysis using the REST API, use the PutBucketAnalyticsConfiguration. You can also use the equivalent operation with the AWS CLI or AWS SDKs.

You can use the following REST APIs to work with Storage Class Analysis:

- DELETE Bucket Analytics configuration
- GET Bucket Analytics configuration
- List Bucket Analytics Configuration

Managing storage costs with Amazon S3 Intelligent-Tiering

The S3 Intelligent-Tiering storage class is designed to optimize storage costs by automatically moving data to the most cost-effective access tier when access patterns change, without operational overhead or impact on performance. For a small monthly object monitoring and automation charge, S3 Intelligent-Tiering monitors access patterns and automatically moves objects that have not been accessed to lower-cost access tiers.

S3 Intelligent-Tiering delivers automatic storage cost savings in three low latency and high throughput access tiers. For data that can be accessed asynchronously, you can choose to activate automatic archiving capabilities within the S3 Intelligent-Tiering storage class. There are no retrieval charges in S3 Intelligent-Tiering. If an object in the Infrequent Access tier or Archive Instant Access tier is accessed later, it is automatically moved back to the Frequent Access tier. No additional tiering charges apply when objects are moved between access tiers within the S3 Intelligent-Tiering storage class.

S3 Intelligent-Tiering is the recommended storage class for data with unknown, changing, or unpredictable access patterns, independent of object size or retention period, such as data lakes, data analytics, and new applications.

The S3 Intelligent-Tiering storage class supports all Amazon S3 features, including the following:

- S3 Inventory, for verifying the access tier of objects
- S3 Replication, for replicating data to any AWS Region
- S3 Storage Lens, for viewing storage usage and activity metrics
- Server-side encryption, for protecting object data
- S3 Object Lock, for preventing accidental deletion of data
- AWS PrivateLink, for accessing Amazon S3 through a private endpoint in a virtual private cloud (VPC)

For information about using S3 Intelligent-Tiering, see the following sections:

Topics

- How S3 Intelligent-Tiering works
- Using S3 Intelligent-Tiering
- Managing S3 Intelligent-Tiering

How S3 Intelligent-Tiering works

The Amazon S3 Intelligent-Tiering storage class automatically stores objects in three access tiers. One tier is optimized for frequent access, one lower-cost tier is optimized for infrequent access, and another very low-cost tier is optimized for rarely accessed data. For a low monthly object monitoring and automation charge, S3 Intelligent-Tiering monitors access patterns and automatically moves objects to the Infrequent Access tier when they haven't been accessed for 30 consecutive days. After 90 days of no access, the objects are moved to the Archive Instant Access tier without performance impact or operational overhead.

To get the lowest storage cost for data that can be accessed in minutes to hours, activate archiving capabilities to add two additional access tiers. You can tier down objects to the Archive Access tier, the Deep Archive Access tier, or both. With Archive Access, S3 Intelligent-Tiering moves objects that have not been accessed for a minimum of 90 consecutive days to the Archive Access tier. With Deep Archive Access, S3 Intelligent-Tiering moves objects to the Deep Archive Access tier after a minimum of 180 consecutive days of no access. For both tiers, you can configure the number of days of no access based on your needs.

The following actions constitute access that prevents tiering your objects down to the Archive Access tier or the Deep Archive Access tier:

- Downloading or copying an object through the Amazon S3 console.
- Invoking CopyObject, UploadPartCopy, or replicating objects with S3 Batch Replication. In these cases, the source objects of the copy or replication operations are tiered up.
- Invoking GetObject, PutObject, RestoreObject, CompleteMultipartUpload, or SelectObjectContent.

For example, if your objects are accessed through `SelectObjectContent` before your specified number of days of no access (for example, 180 days), that action resets the timer. Your objects won't move to the Archive Access tier or the Deep Archive Access tier until the time after the last `SelectObjectContent` request reaches your specified number of days.

If an object in the Infrequent Access tier or Archive Instant Access tier is accessed later, it is automatically moved back to the Frequent Access tier.

The following actions constitute access that automatically moves objects from the Infrequent Access tier or the Archive Instant Access tier back to the Frequent Access tier:

- Downloading or copying an object through the Amazon S3 console.
- Invoking CopyObject, UploadPartCopy, or replicating objects with Batch Replication. In these cases, the source objects of the copy or replication operations are tiered up.
- Invoking GetObject, PutObject, RestoreObject, or CompleteMultipartUpload.

Other actions **don't** constitute access that automatically moves objects from the Infrequent Access tier or the Archive Instant Access tier back to the Frequent Access tier. The following is a sample, not a definitive list, of such actions:

- Invoking HeadObject, GetObjectTagging, PutObjectTagging, ListObjects, ListObjectsV2, ListObjectVersions, and UpdateObjectEncryption.
- Invoking SelectObjectContent doesn't constitute access that tiers objects up to a Frequent Access tier. In addition, it doesn't prevent tiering objects down from the Frequent Access tier to the Infrequent Access tier, and then to the Archive Instant Access tier.

You can use S3 Intelligent-Tiering as your default storage class for newly created data by specifying `INTELLIGENT-TIERING` in the x-amz-storage-class request header when calling the `PutObject`, `CopyObject`, or `CreateMultipartUpload` operations. S3 Intelligent-Tiering is designed for 99.9% availability and 99.999999999% durability.

> **Note**
>
> If the size of an object is less than 128 KB, it is not monitored and is not eligible for automatic tiering. Smaller objects are always stored in the Frequent Access tier.

S3 Intelligent-Tiering access tiers

The following section explains the different automatic and optional access tiers. When objects move between access tiers, the storage class remains the same (S3 Intelligent-Tiering).

Frequent Access tier (automatic)

This is the default access tier that any object created or transitioned to S3 Intelligent-Tiering begins its lifecycle in. An object remains in this tier as long as it is being accessed. The Frequent Access tier provides low latency and high-throughput performance.

Infrequent Access tier (automatic)

If an object is not accessed for 30 consecutive days, the object moves to the Infrequent Access tier. The Infrequent Access tier provides low latency and high-throughput performance.

Archive Instant Access tier (automatic)

If an object is not accessed for 90 consecutive days, the object moves to the Archive Instant Access tier. The Archive Instant Access tier provides low latency and high-throughput performance.

Archive Access tier (optional)

S3 Intelligent-Tiering provides you with the option to activate the Archive Access tier for data that can be accessed asynchronously. After activation, the Archive Access tier automatically archives objects that have not been accessed for a minimum of 90 consecutive days. You can extend the last access time for archiving to a maximum of 730 days. The Archive Access tier has the same performance as the S3 Glacier Flexible Retrieval storage class.

Standard retrieval times for this access tier can range from 3–5 hours. If you initiate your restore request by using S3 Batch Operations, your restore starts within minutes. For more information about retrieval options and times, see the section called “Restoring objects from the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers”.

> **Note**
>
> Only activate the Archive Access tier for 90 days if you want to bypass the Archive Instant Access tier. The Archive Access tier delivers slightly lower storage costs, with minute-to-hour retrieval times. The Archive Instant Access tier delivers millisecond access and high-throughput performance.

Deep Archive Access tier (optional)

S3 Intelligent-Tiering provides you with the option to activate the Deep Archive Access tier for data that can be accessed asynchronously. After activation, the Deep Archive Access tier

automatically archives objects that have not been accessed for a minimum of 180 consecutive days. You can extend the last access time for archiving to a maximum of 730 days. The Deep Archive Access tier has the same performance as the S3 Glacier Deep Archive storage class.

Standard retrieval of objects in this access tier occurs within 12 hours. If you initiate your restore request by using S3 Batch Operations, your restore starts within 9 hours. For more information about retrieval options and times, see the section called "Restoring objects from the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers".

Note

Activate the Archive Access and Deep Archive Access tiers only if your objects can be accessed asynchronously by your application. If the object that you are retrieving is stored in the Archive Access or Deep Archive Access tiers, you must first restore the object by using the `RestoreObject` operation.
You can restore archived objects with up to 1,000 transactions per second (TPS) of object restore requests per account per AWS Region from S3 Intelligent-Tiering Archive Access, and S3 Intelligent-Tiering Deep Archive Access.

Using S3 Intelligent-Tiering

You can use the S3 Intelligent-Tiering storage class to automatically optimize storage costs. S3 Intelligent-Tiering delivers automatic cost savings by moving data on a granular object level between access tiers when access patterns change. For data that can be accessed asynchronously, you can choose to enable automatic archiving within the S3 Intelligent-Tiering storage class using the AWS Management Console, AWS CLI, or Amazon S3 API.

Moving data to S3 Intelligent-Tiering

There are two ways to move data into S3 Intelligent-Tiering. You can upload objects directly into S3 Intelligent-Tiering from the console or programmatically using a PUT operation. For more information, see Setting the storage class of an object. You can also configure S3 Lifecycle configurations to transition objects from S3 Standard or S3 Standard-Infrequent Access to S3 Intelligent-Tiering.

Uploading data to S3 Intelligent-Tiering using Direct PUT

When you upload an object to the S3 Intelligent-Tiering storage class using the PUT API operation, you specify S3 Intelligent-Tiering in the `x-amz-storage-class` request header.

The following request stores the image, `my-image.jpg`, in the `myBucket` bucket. The request uses the `x-amz-storage-class` header to request that the object is stored using the S3 Intelligent-Tiering storage class.

Example

```
PUT /my-image.jpg HTTP/1.1
Host: myBucket.s3.<Region>.amazonaws.com (http://amazonaws.com/)
Date: Wed, 1 Sep 2021 17:50:00 GMT
Authorization: authorization string
Content-Type: image/jpeg
Content-Length: 11434
Expect: 100-continue
x-amz-storage-class: INTELLIGENT_TIERING
```

Transitioning data to S3 Intelligent-Tiering from S3 Standard or S3 Standard-Infrequent Access using S3 Lifecycle

You can add rules to an S3 Lifecycle configuration to tell Amazon S3 to transition objects from one storage class to another. For information on supported transitions and related constraints, see Transitioning objects using S3 Lifecycle.

You can specify S3 Lifecycle configurations at the bucket or prefix level. In this S3 Lifecycle configuration rule, the filter specifies a key prefix (`documents/`). Therefore, the rule applies to objects with key name prefix `documents/`, such as `documents/doc1.txt` and `documents/doc2.txt`. The rule specifies a `Transition` action directing Amazon S3 to transition objects to the S3 Intelligent-Tiering storage class 0 days after creation. In this case, objects are eligible for transition to S3 Intelligent-Tiering at midnight UTC following creation.

Example

```
<LifecycleConfiguration>
  <Rule>
    <ID>ExampleRule</ID>
    <Filter>
       <Prefix>documents/</Prefix>
```

```
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>INTELLIGENT_TIERING</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

A versioning-enabled bucket maintains one current object version, and zero or more noncurrent object versions. You can define separate Lifecycle rules for current and noncurrent object versions.

For more information, see Lifecycle configuration elements.

Enabling S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers

To get the lowest storage cost on data that can be accessed in minutes to hours, you can activate one or both of the archive access tiers by creating a bucket, prefix, or object tag level configuration using the AWS Management Console, AWS CLI, or Amazon S3 API.

Using the S3 console

To enable S3 Intelligent-Tiering automatic archiving

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you want.
3. Choose **Properties**.
4. Navigate to the **S3 Intelligent-Tiering Archive configurations** section and choose **Create configuration**.
5. In the **Archive configuration settings** section, specify a descriptive configuration name for your S3 Intelligent-Tiering Archive configuration.
6. Under **Choose a configuration scope**, choose a configuration scope to use. Optionally, you can limit the configuration scope to specified objects within a bucket using a shared prefix, object tag, or combination of the two.

    a. To limit the scope of the configuration, select **Limit the scope of this configuration using one or more filters**.

    b. To limit the scope of the configuration using a single prefix, enter the prefix under **Prefix**.

c. To limit the scope of the configuration using object tags, select **Add tag** and enter a value for Key.

7. Under **Status**, select **Enable**.
8. In the **Archive settings** section, select one or both of the Archive Access tiers to enable.
9. Choose **Create**.

Using the AWS CLI

You can use the following AWS CLI commands to manage S3 Intelligent-Tiering configurations:

- `delete-bucket-intelligent-tiering-configuration`
- `get-bucket-intelligent-tiering-configuration`
- `list-bucket-intelligent-tiering-configurations`
- `put-bucket-intelligent-tiering-configuration`

For instructions on setting up the AWS CLI, see Developing with Amazon S3 using the AWS CLI in the *Amazon S3 API Reference*.

When using the AWS CLI, you cannot specify the configuration as an XML file. You must specify the JSON instead. The following is an example XML S3 Intelligent-Tiering configuration and equivalent JSON that you can specify in an AWS CLI command.

The following example puts an S3 Intelligent-Tiering configuration to the specified bucket.

Example `put-bucket-intelligent-tiering-configuration`

JSON

```
{
  "Id": "string",
  "Filter": {
    "Prefix": "string",
    "Tag": {
      "Key": "string",
      "Value": "string"
    },
    "And": {
      "Prefix": "string",
```

```
      "Tags": [
        {
          "Key": "string",
          "Value": "string"
        }
        ...
      ]
    }
  },
  "Status": "Enabled"|"Disabled",
  "Tierings": [
    {
      "Days": integer,
      "AccessTier": "ARCHIVE_ACCESS"|"DEEP_ARCHIVE_ACCESS"
    }
    ...
  ]
}
```

XML

```
PUT /?intelligent-tiering&id=Id HTTP/1.1
Host: Bucket.s3.amazonaws.com
<?xml version="1.0" encoding="UTF-8"?>
<IntelligentTieringConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Id>string</Id>
   <Filter>
      <And>
         <Prefix>string</Prefix>
         <Tag>
            <Key>string</Key>
            <Value>string</Value>
         </Tag>
         ...
      </And>
      <Prefix>string</Prefix>
      <Tag>
         <Key>string</Key>
         <Value>string</Value>
      </Tag>
   </Filter>
   <Status>string</Status>
   <Tiering>
```

```
      <AccessTier>string</AccessTier>
      <Days>integer</Days>
   </Tiering>
   ...
</IntelligentTieringConfiguration>
```

Using the PUT API operation

You can use the PutBucketIntelligentTieringConfiguration operation for a specified bucket and up to 1,000 S3 Intelligent-Tiering configurations per bucket. You can define which objects within a bucket are eligible for the archive access tiers using a shared prefix or object tag. Using a shared prefix or object tag allows you to align to specific business applications, workflows, or internal organizations. You also have the flexibility to activate the Archive Access tier, the Deep Archive Access tier, or both.

Managing S3 Intelligent-Tiering

The S3 Intelligent-Tiering storage class delivers automatic storage cost savings in three low-latency and high-throughput access tiers. It also offers optional archive capabilities to help you get the lowest storage costs in the cloud for data that can be accessed in minutes to hours.

Identifying which S3 Intelligent-Tiering access tier objects are stored in

To get a list of your objects and their corresponding metadata, including their S3 Intelligent-Tiering access tier, you can use Amazon S3 Inventory. S3 Inventory provides CSV, ORC, or Parquet output files that list your objects and their corresponding metadata. You can receive these inventory reports on either a daily or weekly basis for an Amazon S3 bucket or a shared prefix. (*Shared prefix* refers to objects that have names that begin with a common string.)

Viewing the archive status of an object within S3 Intelligent-Tiering

To receive notice when an object within the S3 Intelligent-Tiering storage class has moved to either the Archive Access tier or the Deep Archive Access tier, you can set up S3 Event Notifications. For more information, see Enabling event notifications.

Amazon S3 can publish event notifications to an Amazon Simple Notification Service (Amazon SNS) topic, an Amazon Simple Queue Service (Amazon SQS) queue, or an AWS Lambda function. For more information, see Amazon S3 Event Notifications.

The following is an example of a message that Amazon S3 sends to publish an `s3:IntelligentTiering` event. For more information, see the section called “Event message structure”.

```
{
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"IntelligentTiering",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
         "requestParameters":{
            "sourceIPAddress":"s3.amazonaws.com"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"amzn-s3-demo-bucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::amzn-s3-demo-bucket"
            },
            "object":{
               "key":"HappyFace.jpg",
               "size":1024,
               "eTag":"d41d8cd98f00b204e9800998ecf8427e",
            }
         },
         "intelligentTieringEventData":{
            "destinationAccessTier": "ARCHIVE_ACCESS"
            }
      }
```

```
    ]
}
```

You can also use a HEAD object request to view an object's archive status. If an object is stored in the S3 Intelligent-Tiering storage class and is in one of the archive tiers, the HEAD object response shows the current archive tier. To show the archive tier, the request uses the x-amz-archive-status header.

The following HEAD object request returns the metadata of an object (in this case, *my-image.jpg*).

Example

```
HEAD /my-image.jpg HTTP/1.1
Host: bucket.s3.region.amazonaws.com
Date: Wed, 28 Oct 2009 22:32:00 GMT
Authorization: AWS AKIAIOSFODNN7EXAMPLE:02236Q3V0RonhpaBX5sCYVf1bNRuU=
```

You can also use HEAD object requests to monitor the status of a `restore-object` request. If the archive restoration is in progress, the HEAD object response includes the x-amz-restore header.

The following sample HEAD object response shows an object archived by using S3 Intelligent-Tiering with a restore request in progress.

Example

```
HTTP/1.1 200 OK
x-amz-id-2: FSVaTMjrmBp3Izs1NnwBZeu7M19iI8UbxMbi0A8AirHANJBo+hEftBuiESACOMJp
x-amz-request-id: E5CEFCB143EB505A
Date: Fri, 13 Nov 2020 00:28:38 GMT
Last-Modified: Mon, 15 Oct 2012 21:58:07 GMT
ETag: "1accb31fcf202eba0c0f41fa2f09b4d7"
x-amz-storage-class: 'INTELLIGENT_TIERING'
x-amz-archive-status: 'ARCHIVE_ACCESS'
x-amz-restore: 'ongoing-request="true"'
x-amz-restore-request-date: 'Fri, 13 Nov 2020 00:20:00 GMT'
Accept-Ranges: bytes
Content-Type: binary/octet-stream
Content-Length: 300
Server: AmazonS3
```

Restoring objects from the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers

To access objects in the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers, you must initiate a restore request, and then wait until the object is moved into the Frequent Access tier. For more information about archived objects, see the section called "Working with archived objects".

When you restore an object from the Archive Access tier or Deep Archive Access tier, the object moves back into the Frequent Access tier. Afterwards, if the object isn't accessed for 30 consecutive days, it automatically moves into the Infrequent Access tier. Then, after a minimum of 90 consecutive days of no access, the object moves into the Archive Access tier. After a minimum of 180 consecutive days of no access, the object moves into the Deep Archive Access tier. For more information, see the section called "How S3 Intelligent-Tiering works".

You can restore an archived object by using the Amazon S3 console, S3 Batch Operations, the Amazon S3 REST API, the AWS SDKs, or the AWS Command Line Interface (AWS CLI). For more information, see the section called "Working with archived objects".

Understanding S3 Glacier storage classes for long-term data storage

You can use Amazon S3 S3 Glacier storage classes to provide cost-effective solutions to storing long-term data that isn't accessed often. The S3 Glacier storage classes are:

- S3 Glacier Instant Retrieval
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive

You choose one of these storage classes based on how often you access your data and how fast you need to retrieve it. Each of these storage classes offer the same durability and resiliency as the S3 Standard storage class, but at lower storage costs. For more information about the S3 Glacier storage classes, see https://aws.amazon.com/s3/storage-classes/glacier/.

Topics

- Comparing the S3 Glacier storage classes
- S3 Glacier Instant Retrieval
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive

- Understanding archival storage in S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive
- How these storage classes differ from the Amazon Glacier service

Comparing the S3 Glacier storage classes

Each S3 Glacier storage class has a minimum storage duration for all objects. If you delete, overwrite, or transition the object to a different storage class before the minimum, you are charged for the remainder of that duration.

Some S3 Glacier storage classes are archival, which means the objects stored in those classes are archived and not available for real-time access. For more information, see Understanding archival storage in S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive.

Storage classes designed for less frequent access patterns with longer retrieval times offer lower storage costs. For pricing information, see https://aws.amazon.com/s3/pricing/.

The following table summarizes the key points to consider when choosing a S3 Glacier storage class:

S3 Glacier Instant Retrieval

We recommend using S3 Glacier Instant Retrieval for long-term data that's accessed once per quarter and requires millisecond retrieval times. This storage class is ideal for performance-sensitive use cases such as image hosting, file-sharing applications, and storing medical records for access during appointments.

S3 Glacier Instant Retrieval storage class offers real-time access to your objects with the same latency and throughput performance as the S3 Standard-IA storage class. When compared to S3 Standard-IA, S3 Glacier Instant Retrieval has lower storage costs but higher data access costs.

There is a minimum object size of 128 KB for data stored in the S3 Glacier Instant Retrieval storage class. This storage class also has a minimum storage duration period of 90 days.

S3 Glacier Flexible Retrieval

We recommend using S3 Glacier Flexible Retrieval for archive data that's accessed one to two times a year and doesn't require immediate access. S3 Glacier Flexible Retrieval offers flexible retrieval times to help you balance costs, with access times ranging from a few minutes to hours, and free bulk retrievals. This storage class is ideal for backup and disaster recovery.

Objects stored in S3 Glacier Flexible Retrieval are archived and not available for real-time access. For more information, see Understanding archival storage in S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive. To access these objects, you first initiate a restore request which creates a temporary copy of the object that you can access when the request completes. For information, see Working with archived objects. When you restore an object, you can choose a retrieval tier to meet your use case, with lower costs for longer restore times.

The following retrieval tiers are available for S3 Glacier Flexible Retrieval:

- **Expedited retrieval** – Typically restores the object in 1–5 minutes. Expedited retrievals are subject to demand, so to make sure you have reliable and predictable restore times, we recommend that you purchase provisioned retrieval capacity. For more information, see Provisioned capacity.
- **Standard retrieval** – Typically restores the object in 3–5 hours, or within 1 minute to 5 hours when you use S3 Batch Operations. For more information, see Restore objects with Batch Operations.
- **Bulk retrieval** – Typically restores the object within 5–12 hours. Bulk retrievals are free.

For any retrieval option, objects larger than 5 TB typically finish within 48 hours with up to 300 megabytes per second (MBps) of retrieval throughput. For more information, see Understanding archive retrieval options.

The minimum storage duration for objects in S3 Glacier Flexible Retrieval storage class is 90 days.

S3 Glacier Flexible Retrieval requires 40 KB of additional metadata for each object. This includes 32 KB of metadata required to identify and retrieve your data, which is charged at the default rate for S3 Glacier Flexible Retrieval. An additional 8 KB data is required to maintain the user-defined name and metadata for archived objects, and is charged at the S3 Standard rate.

S3 Glacier Deep Archive

We recommend using S3 Glacier Deep Archive for archive data that's accessed less than once a year. This storage class is designed for retaining data sets for multiple years to meet compliance requirements and can also be used for backup or disaster recovery or any infrequently accessed data that you can wait up to 72 hours to retrieve. S3 Glacier Deep Archive is the lowest-cost storage option in AWS.

Objects stored in S3 Glacier Deep Archive are archived and not available for real-time access. For more information, see Understanding archival storage in S3 Glacier Flexible Retrieval and S3

Glacier Deep Archive. To access these objects, you first initiate a restore request which creates a temporary copy of the object that you can access when the request completes. For information, see Working with archived objects. When you restore an object, you can choose a retrieval tier to meet your use case, with lower costs for longer restore times.

The following retrieval tiers are available for S3 Glacier Deep Archive:

- **Standard retrieval** – Typically restores the object within 12 hours, or within 9–12 hours when you use S3 Batch Operations. For more information, see Restore objects with Batch Operations.
- **Bulk retrieval** – Typically restores the object within 48 hours at a fraction of the cost of the Standard retrieval tier.

The minimum storage duration for objects in S3 Glacier Deep Archive storage class is 180 days.

S3 Glacier Deep Archive requires 40 KB of additional metadata for each object. This includes 32 KB of metadata required to identify and retrieve your data, which is charged at the default rate for S3 Glacier Deep Archive. An additional 8 KB data is required to maintain the user-defined name and metadata for archived objects, and is charged at the S3 Standard rate.

Understanding archival storage in S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive

S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive are archival storage classes. This means that when you store an object in these storage classes that object is archived, and cannot be accessed directly. To access an archived object, you submit a restore request for it, and then wait for the service to restore the object. The restore request restores a temporary copy of the object, and that copy is deleted when the duration you specified in the request expires. For more information, see Working with archived objects.

The transition of objects to the S3 Glacier Deep Archive storage class can go only one way.

If you want to change the storage class of an archived object to another storage class, you must use the restore operation to make a temporary copy of the object first. Then use the copy operation to overwrite the object specifying S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or Reduced Redundancy Storage as the storage class.

> **Note**
>
> The Copy operation for restored objects isn't supported in the Amazon S3 console for objects in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. For this type of Copy operation, use the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the REST API.

You can restore archived objects in these storage classes with up to 1,000 transactions per second (TPS) of object restore requests per account per AWS Region.

Cost considerations

If you are planning to archive infrequently accessed data for a period of months or years, the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes can reduce your storage costs. However, to ensure that the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage class is appropriate for you, consider the following:

- **Storage overhead charges** – Each archived object requires 40 KB of additional metadata. This includes 32 KB of metadata required to identify and retrieve your data, which is charged at the default rate for that storage class. An additional 8 KB data is required to maintain the user-defined name and metadata for archived objects, and is charged at the S3 Standard rate.

  If you are archiving small objects, consider these storage charges. Also consider aggregating many small objects into a smaller number of large objects to reduce overhead costs.
- **Multipart upload pricing** – Objects in S3-storage-class-glacier; and S3 Glacier Deep Archive are billed at S3 Standard storage class rates when you upload them using multipart uploads. For more information, see Multipart upload and pricing.
- **Minimum 30 day storage charges** – S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive are long-term archival solutions. The minimal storage duration period is 90 days for the S3 Glacier Flexible Retrieval storage class and 180 days for S3 Glacier Deep Archive. Deleting data that is archived to these storage classes doesn't incur charges if the objects you delete are archived for more than the minimal storage duration period. If you delete or overwrite an archived object within the minimal duration period, Amazon S3 charges for the remainder of that duration.
- **Data retrieval charges** – When you restore an archived objects to S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive there are per-request data retrieval charges. These charges vary based on the retrieval tier you choose when you initiate a restore. For pricing information, see Amazon S3 pricing.

- **S3 Lifecycle** – When you restore an archived objects to S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive there are per-request data retrieval charges. These charges vary based on the retrieval tier you choose when you initiate a restore. For pricing information, see Amazon S3 pricing.

Restoring archived objects

Archived objects aren't accessible in real time. You must first initiate a restore request and then wait until a temporary copy of the object is available for the duration that you specify in the request. After you receive a temporary copy of the restored object, the object's storage class remains S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive. (A HeadObject or GetObject API operation request will return S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive as the storage class.)

> **Note**
>
> When you restore an archive, you are paying for both the archive (S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive rate) and a copy that you restored temporarily (S3 Standard storage rate). For information about pricing, see Amazon S3 pricing.

You can restore an object copy programmatically or by using the Amazon S3 console. Amazon S3 processes only one restore request at a time per object. For more information, see Restoring an archived object.

How these storage classes differ from the Amazon Glacier service

The S3 Glacier storage classes are part of the Amazon S3 service and store data as objects in S3 buckets. You can manage objects in these storage classes using the S3 console or programmatically using the S3 APIs or SDKs. When you store objects in S3 Glacier storage classes, you can use S3 features such as advanced encryption, object tagging, and S3 Lifecycle configurations to help manage data accessibility and cost.

> **Important**
>
> We recommend using the S3 Glacier storage classes within the Amazon S3 service for all of your long-term data.

The Amazon Glacier service is a separate service that stores data as archives within vaults. This service doesn't support Amazon S3 features and doesn't provide console support for data upload and download operations. We don't recommend using the Amazon Glacier service for your long-term data. Data stored in this service isn't accessible from the Amazon S3 service. If you are looking for information on the Amazon Glacier service, see the Amazon Glacier Developer Guide. To transfer data from the Amazon Glacier service to a storage class in Amazon S3 see Data Transfer from Amazon Glacier Vaults to Amazon S3 in the AWS solutions library.

Working with archived objects

To reduce your storage costs for infrequently accessed objects, you can *archive* those objects. When you archive an object, it is moved into low-cost storage, which means that you can't access it in real time.

Although archived objects are not accessible in real time, you can restore them in minutes or hours, depending on the storage class. You can restore an archived object by using the Amazon S3 console, S3 Batch Operations, the REST API, the AWS SDKs, and the AWS Command Line Interface (AWS CLI). For instructions, see Restoring an archived object.

Amazon S3 objects in the following storage classes or tiers are archived and are not accessible in real time:

- The S3 Glacier Flexible Retrieval storage class
- The S3 Glacier Deep Archive storage class
- The S3 Intelligent-Tiering Archive Access tier
- The S3 Intelligent-Tiering Deep Archive Access tier

> **Note**
>
> Objects stored in the S3 Glacier Instant Retrieval storage class are not archived. S3 Glacier Instant Retrieval provides real-time access with millisecond retrieval latency and does not require a restore operation. For more information about storage classes, see Understanding and managing Amazon S3 storage classes.

To restore archived objects, you must do the following:

- For objects in the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes, you must initiate the restore request and wait until a temporary copy of the object is available. When a temporary copy of the restored object is created, the object's storage class remains the same. (A HeadObject or GetObject API operation request returns S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive as the storage class.)
- For objects in the S3 Intelligent-Tiering Archive Access and S3 Intelligent-Tiering Deep Archive Access tiers, you must initiate the restore request and wait until the object is moved into the Frequent Access tier.

For more information about how all Amazon S3 storage classes compare, see Understanding and managing Amazon S3 storage classes. For more information about S3 Intelligent-Tiering, see the section called "How S3 Intelligent-Tiering works".

The time it takes a restore job to finish depends on which archive storage class or storage tier you use and which retrieval option you specify: Expedited (only available for S3 Glacier Flexible Retrieval and S3 Intelligent-Tiering Archive Access), Standard, or Bulk. For more information, see Understanding archive retrieval options.

You can be notified when your restore is complete by using Amazon S3 Event Notifications. For more information, see Amazon S3 Event Notifications.

Restoring objects from Amazon Glacier

When you use S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, Amazon S3 restores a temporary copy of the object only for the specified duration. After that, it deletes the restored object copy. You can modify the expiration period of a restored copy by reissuing a restore request. In this case, Amazon S3 updates the expiration period relative to the current time.

> **Note**
>
> When you restore an archived object from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, you pay for both the archived object and the copy that you restored temporarily. For information about pricing, see Amazon S3 pricing.

Amazon S3 calculates the expiration time of the restored object copy by adding the number of days specified in the restoration request to the time when the requested restoration is completed. Amazon S3 then rounds the resulting time to the next day at midnight Universal Coordinated Time

(UTC). For example, suppose that a restored object copy was created on October 15, 2012, at 10:30 AM UTC, and the restoration period was specified as 3 days. In this case, the restored copy expires on October 19, 2012, at 00:00 UTC, at which time Amazon S3 deletes the object copy.

Restoring objects from S3 Intelligent-Tiering

When you restore an object from the S3 Intelligent-Tiering Archive Access tier or S3 Intelligent-Tiering Deep Archive Access tier, the object moves back into the S3 Intelligent-Tiering Frequent Access tier. If the object is not accessed after 30 consecutive days, it automatically moves into the Infrequent Access tier. After a minimum of 90 consecutive days of no access, the object moves into the S3 Intelligent-Tiering Archive Access tier. If the object is not accessed after a minimum of 180 consecutive days, the object moves into the Deep Archive Access tier.

> **Note**
>
> Unlike in the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes, restore requests for S3 Intelligent-Tiering objects don't accept the `Days` value.

Using S3 Batch Operations with restore requests

To restore more than one Amazon S3 object with a single request, you can use S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on. S3 Batch Operations calls the respective API operation to perform the specified operation. A single Batch Operations job can perform the specified operation on billions of objects containing exabytes of data.

Topics

- Understanding archive retrieval options
- Restoring an archived object

Understanding archive retrieval options

Amazon S3 has three archival storage classes - S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, and S3 Glacier Deep Archive. While objects stored in the S3 Glacier Instant Retrieval storage class are immediately available using GET, to access data stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes you first need to retrieve data using the RestoreObject REST API. Restoring datasets made up of tens of millions of objects or hundreds of

terabytes of data could take longer than typical restore times and need special considerations. For more information, see Restoring large datasets.

You can choose from three retrieval access options to restore your archived objects based on your desired retrieval speed – Expedited, Standard and Bulk.

- **Expedited retrieval** – Quickly access your data stored in the S3 Glacier Flexible Retrieval storage class or S3 Intelligent-Tiering Archive Access tier. You can use this option for occasional urgent requests for up to hundreds of objects. Objects under 250 megabytes in size are typically made available within 1–5 minutes, and objects 250 megabytes or larger in size are typically retrieved with up to 300 megabytes per second of retrieval throughput. In addition, you have the option to purchase Provisioned Capacity for Expedited retrievals. Provisioned Capacity helps ensure that Expedited retrieval capacity is available when you need it. For more information, see Provisioned capacity.

  > **Note**
  >
  > Expedited retrievals are a premium feature and are charged at the Expedited request and retrieval rate. For information about Amazon S3 pricing, see Amazon S3 Pricing.

- **Standard retrieval** – Access your data within several hours. Standard is the default option for requests that do not specify the retrieval option. Standard retrievals typically finish within 3–5 hours for the S3 Glacier Flexible Retrieval storage class or S3 Intelligent-Tiering Archive Access tier. Standard retrievals typically finish within 12 hours for the S3 Glacier Deep Archive storage class or S3 Intelligent-Tiering Deep Archive Access tier. Standard retrievals are free for objects stored in the S3 Intelligent-Tiering storage class.

  > **Note**
  >
  > - For objects stored in the S3 Glacier Flexible Retrieval storage class or the S3 Intelligent-Tiering Archive Access tier, Standard retrievals initiated by using the S3 Batch Operations restore operation typically start within minutes and finish within 3-5 hours at a throughput of up to 1–2 petabytes per day.
  > - For objects in the S3 Glacier Deep Archive storage class or the S3 Intelligent-Tiering Deep Archive Access tier, Standard retrievals initiated by using Batch Operations typically start to complete within 9 hours at a throughput of up to 1–2 petabytes per day.

- **Bulk retrieval** – Access your data by using the lowest-cost retrieval option in S3 Glacier storage classes. With Bulk retrievals, you can retrieve large amounts of data inexpensively. For objects stored in the S3 Glacier Flexible Retrieval storage class or the S3 Intelligent-Tiering Archive Access tier, Bulk retrievals typically finish within 5–12 hours. For objects stored in the S3 Glacier Deep Archive storage class or the S3 Intelligent-Tiering Deep Archive Access tier, these retrievals typically finish within 48 hours. Bulk retrievals are free for objects that are stored in the S3 Glacier Flexible Retrieval or S3 Intelligent-Tiering storage classes.

The following table summarizes the archive retrieval options. For information about pricing, see Amazon S3 pricing.

To make an `Expedited`, `Standard`, or `Bulk` retrieval, set the `Tier` request element in the RestoreObject REST API operation request to the option that you want, or the equivalent in the AWS Command Line Interface (AWS CLI) or AWS SDKs. If you purchased provisioned capacity, all Expedited retrievals are automatically served through your provisioned capacity.

Restoring large datasets

Restoring datasets made up of tens of millions of objects or hundreds of terabytes of data could take longer than typical restore times for any retrieval tier due to retrieval limits.

When you initiate restore requests for objects that are stored in the S3 Glacier Flexible Retrieval, S3 Glacier Deep Archive, or S3 Intelligent-Tiering storage classes, a retrieval-requests quota is applied for your AWS account. S3 Glacier supports restore requests at a rate of 1,000 transactions per second. If this rate is exceeded otherwise valid requests are throttled or rejected and Amazon S3 returns a `ThrottlingException` error. You can use S3 Batch Operations to retrieve many objects with a single request, which fully utilizes the restore request rate available in your account. For more information, see Performing object operations in bulk with Batch Operations.

After you initiate restore requests, S3 Glacier supports restoring large datasets at a throughput of up to 1–2 petabytes per day per customer account. For any retrieval option, objects larger than 5 terabytes will take longer to be restored with up to 300 megabytes per second of retrieval throughput. For example, a 50-terabyte S3 Glacier Flexible Retrieval object could take up to 48 hours to complete. If you require increased restoration limits, you can contact AWS Support.

Provisioned capacity

Provisioned capacity helps ensure that your retrieval capacity for Expedited retrievals from S3 Glacier Flexible Retrieval is available when you need it. Each unit of capacity provides that at least

three Expedited retrievals can be performed every 5 minutes, and it provides up to 300 megabytes per second of retrieval throughput.

Without provisioned capacity, Expedited retrievals might not be accepted during periods of high demand. For predictable, fast access to more of your data, consider using the S3 Glacier Instant Retrieval storage class.

Provisioned capacity units are allocated to an AWS account. Thus, the requester of the Expedited data retrieval should purchase the provisioned capacity unit, not the bucket owner.

You can purchase provisioned capacity by using the Amazon S3 console, the Amazon Glacier console, the Purchase Provisioned Capacity REST API operation, the AWS SDKs, or the AWS CLI. For provisioned capacity pricing information, see Amazon S3 pricing.

Restoring an archived object

Amazon S3 objects in the following storage classes or tiers are archived and are not accessible in real time:

- The S3 Glacier Flexible Retrieval storage class
- The S3 Glacier Deep Archive storage class
- The S3 Intelligent-Tiering Archive Access tier
- The S3 Intelligent-Tiering Deep Archive Access tier

Amazon S3 objects that are stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes are not immediately accessible. To access an object in these storage classes, you must restore a temporary copy of the object to its S3 bucket for a specified duration (number of days). If you want a permanent copy of the object, restore the object, and then create a copy of it in your Amazon S3 bucket. Copying restored objects isn't supported in the Amazon S3 console. For this type of copy operation, use the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the REST API. Unless you make a copy and change its storage class, the object will still be stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. For information about using these storage classes, see Storage classes for rarely accessed objects.

To access objects in the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers, you must initiate a restore request and wait until the object is moved into the Frequent Access tier. When you restore an object from the Archive Access tier or Deep Archive Access tier, the object

moves back into the Frequent Access tier. For information about using these storage classes, see Storage class for automatically optimizing data with changing or unknown access patterns.

For general information about archived objects, see Working with archived objects.

> **Note**
>
> - When you restore an archived object from the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes, you pay for both the archived object and the copy that you restored temporarily.
> - When you restore an object from S3 Intelligent-Tiering, there are no retrieval charges for Standard or Bulk retrievals.
> - Subsequent restore requests called on archived objects that have already been restored are billed as GET requests. For information about pricing, see Amazon S3 pricing.

Restoring an archived object

You can restore an archived object by using the Amazon S3 console, the Amazon S3 REST API, the AWS SDKs, the AWS Command Line Interface (AWS CLI), or S3 Batch Operations.

Using the S3 console

Restore objects using the Amazon S3 console

Use the following procedure to Restore an object that has been archived to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes, or the S3 Intelligent-Tiering Archive Access or Deep Archive Access storage tiers.

To restore an archived object

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that contains the objects that you want to restore.
4. In the **Objects** list, select the object or objects that you want to restore, choose **Actions**, and then choose **Initiate restore**.

5. If you're restoring from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, enter the number of days that you want your archived data to be accessible in the **Number of days that the restored copy is available** box.
6. For **Retrieval tier**, do one of the following:

   - Choose **Bulk retrieval** or **Standard retrieval**, and then choose **Initiate restore**.
   - Choose **Expedited retrieval** (available only for S3 Glacier Flexible Retrieval or S3 Intelligent-Tiering Archive Access). If you're restoring an object in S3 Glacier Flexible Retrieval, you can choose whether to buy provisioned capacity for your Expedited retrieval. If you want to purchase provisioned capacity, proceed to the next step. If you don't, choose **Initiate restore**.

   > **Note**
   >
   > Objects from the S3 Intelligent-Tiering Archive Access and Deep Archive Access tiers are automatically restored to the Frequent Access tier.

7. (Optional) If you're restoring an object in S3 Glacier Flexible Retrieval and you chose **Expedited retrieval**, you can choose whether to buy provisioned capacity. Provisioned capacity is available only for objects in S3 Glacier Flexible Retrieval. If you already have provisioned capacity, choose **Initiate restore** to start a provisioned retrieval.

   If you have provisioned capacity, all of your Expedited retrievals are served by your provisioned capacity. For more information, see Provisioned capacity.

   - If you don't have provisioned capacity and you don't want to buy it, choose **Initiate restore**.
   - If you don't have provisioned capacity, but you want to buy provisioned capacity units (PCUs), choose **Purchase PCUs**. In the **Purchase PCUs** dialog box, choose how many PCUs you want to buy, confirm your purchase, and then choose **Purchase PCUs**. When you get the **Purchase succeeded** message, choose **Initiate restore** to start provisioned retrieval.

Using the AWS CLI

Restore objects from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive

The following example uses the `restore-object` command to restore the object *`dir1/example.obj`* in the bucket *`amzn-s3-demo-bucket`* for 25 days.

```
aws s3api restore-object --bucket amzn-s3-demo-bucket --key dir1/example.obj --restore-request '{"Days":25,"GlacierJobParameters":{"Tier":"Standard"}}'
```

If the JSON syntax used in the example results in an error on a Windows client, replace the restore request with the following syntax:

```
--restore-request Days=25,GlacierJobParameters={"Tier"="Standard"}
```

Restore objects from S3 Intelligent-Tiering Archive Access and Deep Archive Access

The following example uses the `restore-object` command to restore the object *dir1/example.obj* in the bucket *amzn-s3-demo-bucket* to the Frequent Access tier.

```
aws s3api restore-object --bucket amzn-s3-demo-bucket --key dir1/example.obj --restore-request '{}'
```

> **Note**
>
> Unlike in the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes, restore requests for S3 Intelligent-Tiering objects don't accept the `Days` value.

Monitor restore status

To monitor the status of your `restore-object` request, use the following `head-object` command:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key dir1/example.obj
```

For more information, see restore-object in the *AWS CLI Command Reference*.

Using the REST API

Amazon S3 provides an API operation for you to initiate the restoration of an archived object. For more information, see RestoreObject in the *Amazon Simple Storage Service API Reference*.

Using the AWS SDKs

For examples of how to restore archived objects in S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive with the AWS SDKs, see Code examples in the *Amazon S3 API Reference*.

Using S3 Batch Operations

To restore more than one archived object with a single request, you can use S3 Batch Operations. You provide S3 Batch Operations with a list of objects to operate on. S3 Batch Operations calls the respective API operation to perform the specified operation. A single Batch Operations job can perform the specified operation on billions of objects containing exabytes of data.

To create a Batch Operations job, you must have a manifest that contains only the objects that you want to restore. You can create a manifest by using S3 Inventory, or you can supply a CSV file with the necessary information. For more information, see the section called “Specifying a manifest”.

Before creating and running S3 Batch Operations jobs, you must grant permissions to Amazon S3 to perform S3 Batch Operations on your behalf. For the required permissions, see the section called “Granting permissions”.

> **Note**
>
> Batch Operations jobs can operate either on S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage class objects *or* on S3 Intelligent-Tiering Archive Access and Deep Archive Access storage tier objects. Batch Operations can't operate on both types of archived objects in the same job. To restore objects of both types, you *must* create separate Batch Operations jobs.
>
> For more information about using Batch Operations to restore archive objects, see the section called “Restore objects”.

To create an S3 Initiate Restore Object Batch Operations job

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Batch Operations**.
3. Choose **Create job**.
4. For **AWS Region**, choose the Region where you want to create your job.
5. Under **Manifest format**, choose the type of manifest to use.
   - If you choose **S3 inventory report**, enter the path to the `manifest.json` object that Amazon S3 generated as part of the CSV-formatted inventory report. If you want to use a

manifest version other than the most recent, enter the version ID for the `manifest.json` object.

- If you choose **CSV**, enter the path to a CSV-formatted manifest object. The manifest object must follow the format described in the console. If you want to use a version other than the most recent, you can optionally include the version ID for the manifest object.

6. Choose **Next**.
7. In the **Operation** section, choose **Restore**.
8. In the **Restore** section, for **Restore source**, choose either **Glacier Flexible Retrieval or Glacier Deep Archive** or **Intelligent-Tiering Archive Access tier or Deep Archive Access tier**.

   If you chose **Glacier Flexible Retrieval or Glacier Deep Archive**, enter a number for **Number of days that the restored copy is available**.

   For **Retrieval tier**, choose the tier that you want to use.
9. Choose **Next**.
10. On the **Configure additional options** page, fill out the following sections:

    - In the **Additional options** section, provide a description for the job and specify a priority number for the job. Higher numbers indicate a higher priority. For more information, see the section called "Assigning job priority".
    - In the **Completion report** section, select whether Batch Operations should create a completion report. For more information about completion reports, see the section called "Completion reports".
    - In the **Permissions** section, you must grant permissions to Amazon S3 to perform Batch Operations on your behalf. For the required permissions, see the section called "Granting permissions".
    - (Optional) In the **Job tags** section, add tags in key-value pairs. For more information, see the section called "Using tags".

    When you're finished, choose **Next**.
11. On the **Review** page, verify the settings. If you need to make changes, choose **Previous**. Otherwise, choose **Create job**.

For more information about Batch Operations, see Restore objects with Batch Operations and Creating an S3 Batch Operations job.

Checking the restore status and expiration date

You can check the status of a restore request or the expiration date by using the Amazon S3 console, Amazon S3 Event Notifications, the AWS CLI, or the Amazon S3 REST API.

> **Note**
>
> Objects restored from the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes are stored only for the number of days that you specify. The following procedures return the expiration date for these copies.
> Objects restored from the S3 Intelligent-Tiering Archive Access and Deep Archive Access storage tiers don't have expiration dates and instead are moved back to the Frequent Access tier.

Using the S3 console

To check an object's restore status and expiration date in the Amazon S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that contains the object that you are restoring.
4. In the **Objects** list, select the object that you are restoring. The object's details page appears.

   - If the restoration isn't finished, a section says **Restoration in progress**.
   - If the restoration is finished, a section says **Restoration complete**. If you're restoring from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, this section also displays the **Restoration expiry date**. Amazon S3 will remove the restored copy of your archived object on this date.

Using Amazon S3 Event Notifications

You can be notified of object restoration completion by using the `s3:ObjectRestore:Completed` action with the Amazon S3 Event Notifications feature. For

more information about enabling event notifications, see Enabling notifications by using Amazon SQS, Amazon SNS, and AWS Lambda. For more information about the various `ObjectRestore` event types, see the section called "Supported event types for SQS, SNS, and Lambda".

Using the AWS CLI

Check an object's restore status and expiration date with the AWS CLI

The following example uses the `head-object` command to view metadata for the object *`dir1/example.obj`* in the bucket *`amzn-s3-demo-bucket`*. When you run this command on an object being restored Amazon S3 returns if the restore is ongoing and (if applicable) the expiration date.

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key dir1/example.obj
```

Expected output (restore ongoing):

```
{
    "Restore": "ongoing-request=\"true\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
    "StorageClass": "GLACIER"
}
```

Expected output (restore finished):

```
{
    "Restore": "ongoing-request=\"false\", expiry-date=\"Wed, 12 Aug 2020 00:00:00 GMT
\"",
    "LastModified": "2020-06-16T21:55:22+00:00",
    "ContentLength": 405,
    "ETag": "\"b662d79adeb7c8d787ea7eafb9ef6207\"",
    "VersionId": "wbYaE2vtOV0iIBXrOqGAJt3fP1cHB8Wi",
    "ContentType": "binary/octet-stream",
    "ServerSideEncryption": "AES256",
    "Metadata": {},
    "StorageClass": "GLACIER"
```

```
}
```

For more information about `head-object`, see head-object in the *AWS CLI Command Reference*.

Using the REST API

Amazon S3 provides an API operation for you to retrieve object metadata. To check the restoration status and expiration date of an archived object using the REST API, see HeadObject in the *Amazon Simple Storage Service API Reference*.

Upgrading the speed of an in-progress restore

You can upgrade the speed of your restoration while it is in progress.

To upgrade an in-progress restore to a faster tier

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that contains the objects that you want to restore.
4. In the **Objects** list, select the object that you are restoring. The object's details page appears. On the object's details page, choose **Upgrade retrieval tier**. For information about checking the restoration status of an object, see Checking the restore status and expiration date.
5. Choose the tier that you want to upgrade to, and then choose **Initiate restore**.

Managing the lifecycle of objects

S3 Lifecycle helps you store objects cost effectively throughout their lifecycle by transitioning them to lower-cost storage classes, or, deleting expired objects on your behalf. To manage the lifecycle of your objects, create an *S3 Lifecycle configuration* for your bucket. An S3 Lifecycle configuration is a set of rules that define actions that Amazon S3 applies to a group of objects. There are two types of actions:

- **Transition actions** – These actions define when objects transition to another storage class. For example, you might choose to transition objects to the S3 Standard-IA storage class 30 days after creating them, or archive objects to the S3 Glacier Flexible Retrieval storage class one year after creating them. For more information, see Understanding and managing Amazon S3 storage classes.

There are costs associated with lifecycle transition requests. For pricing information, see Amazon S3 pricing.

- **Expiration actions** – These actions define when objects expire. Amazon S3 deletes expired objects on your behalf. For example, you might to choose to expire objects after they have been stored for a regulatory compliance period. For more information, see Expiring objects.

  There are potential costs associated with lifecycle expiration only when you expire objects in a storage class with a minimum storage duration. For more information, see Minimum storage duration charge.

> **Important**
>
> **General purpose buckets** — You can't use a bucket policy to prevent deletions or transitions by an S3 Lifecycle rule. For example, even if your bucket policy denies all actions for all principals, your S3 Lifecycle configuration still functions as normal.

> **Note**
>
> S3 Lifecycle does not independently manage annotations. Annotation storage is always billed at S3 Standard rates in Frequent Access storage, regardless of the parent object's storage class. When a Lifecycle rule expires an object, all annotations associated with that object version are deleted.

Existing and new objects

When you add a Lifecycle configuration to a bucket, the configuration rules apply to both existing objects and objects that you add later. For example, if you add a Lifecycle configuration rule today with an expiration action that causes objects to expire 30 days after creation, Amazon S3 will queue for removal any existing objects that are more than 30 days old.

Changes in billing

If there is any delay between when an object becomes eligible for a lifecycle action and when Amazon S3 transfers or expires your object, billing changes are applied as soon as the object becomes eligible for the lifecycle action. For example, if an object is scheduled to expire and

Amazon S3 doesn't immediately expire the object, you won't be charged for storage after the expiration time.

The one exception to this behavior is if you have a lifecycle rule to transition to the S3 Intelligent-Tiering storage class. In that case, billing changes don't occur until the object has transitioned to S3 Intelligent-Tiering. For more information about S3 Lifecycle rules, see Lifecycle configuration elements.

> **Note**
>
> There are no data retrieval charges for lifecycle transitions. However, there are per-request ingestion charges when using PUT, COPY, or lifecycle rules to move data into any S3 storage class. Consider the ingestion or transition cost before moving objects into any storage class. For more information about cost considerations, see Amazon S3 pricing.

Monitoring the effect of lifecycle rules

To monitor the effect of updates made by active lifecycle rules, see the section called "How do I monitor the actions taken by my lifecycle rules?".

Managing the complete lifecycle of objects

With S3 Lifecycle configuration rules you can tell Amazon S3 to transition objects to less-expensive storage classes, archive or delete them. For example:

- If you upload periodic logs to a bucket, your application might need them for a week or a month. After that, you might want to delete them.
- Some documents are frequently accessed for a limited period of time. After that, they are infrequently accessed. At some point, you might not need real-time access to them, but your organization or regulations might require you to archive them for a specific period. After that, you can delete them.
- You might upload some types of data to Amazon S3 primarily for archival purposes. For example, you might archive digital media, financial, and healthcare records, raw genomics sequence data, long-term database backups, and data that must be retained for regulatory compliance.

By combining S3 Lifecycle actions to manage an object's complete lifecycle. For example, suppose that the objects you create have a well-defined lifecycle. Initially, the objects are frequently

accessed for a period of 30 days. Then, objects are infrequently accessed for up to 90 days. After that, the objects are no longer needed, so you might choose to archive or delete them.

In this scenario, you can create an S3 Lifecycle rule in which you specify the initial transition action to S3 Intelligent-Tiering, S3 Standard-IA, or S3 One Zone-IA storage, another transition action to S3 Glacier Flexible Retrieval storage for archiving, and an expiration action. As you move the objects from one storage class to another, you save on storage costs. For more information about cost considerations, see Amazon S3 pricing.

Topics

- Transitioning objects using Amazon S3 Lifecycle
- Expiring objects
- Setting an S3 Lifecycle configuration on a bucket
- How S3 Lifecycle interacts with other bucket configurations
- Configuring S3 Lifecycle event notifications
- Lifecycle configuration elements
- How Amazon S3 handles conflicts in lifecycle configurations
- Examples of S3 Lifecycle configurations
- Troubleshooting Amazon S3 Lifecycle issues

Transitioning objects using Amazon S3 Lifecycle

You can add transition actions to an S3 Lifecycle configuration to tell Amazon S3 to move objects to another Amazon S3 storage class. For more information about storage classes, see Understanding and managing Amazon S3 storage classes. Some examples of when you might use S3 Lifecycle configurations in this way include the following:

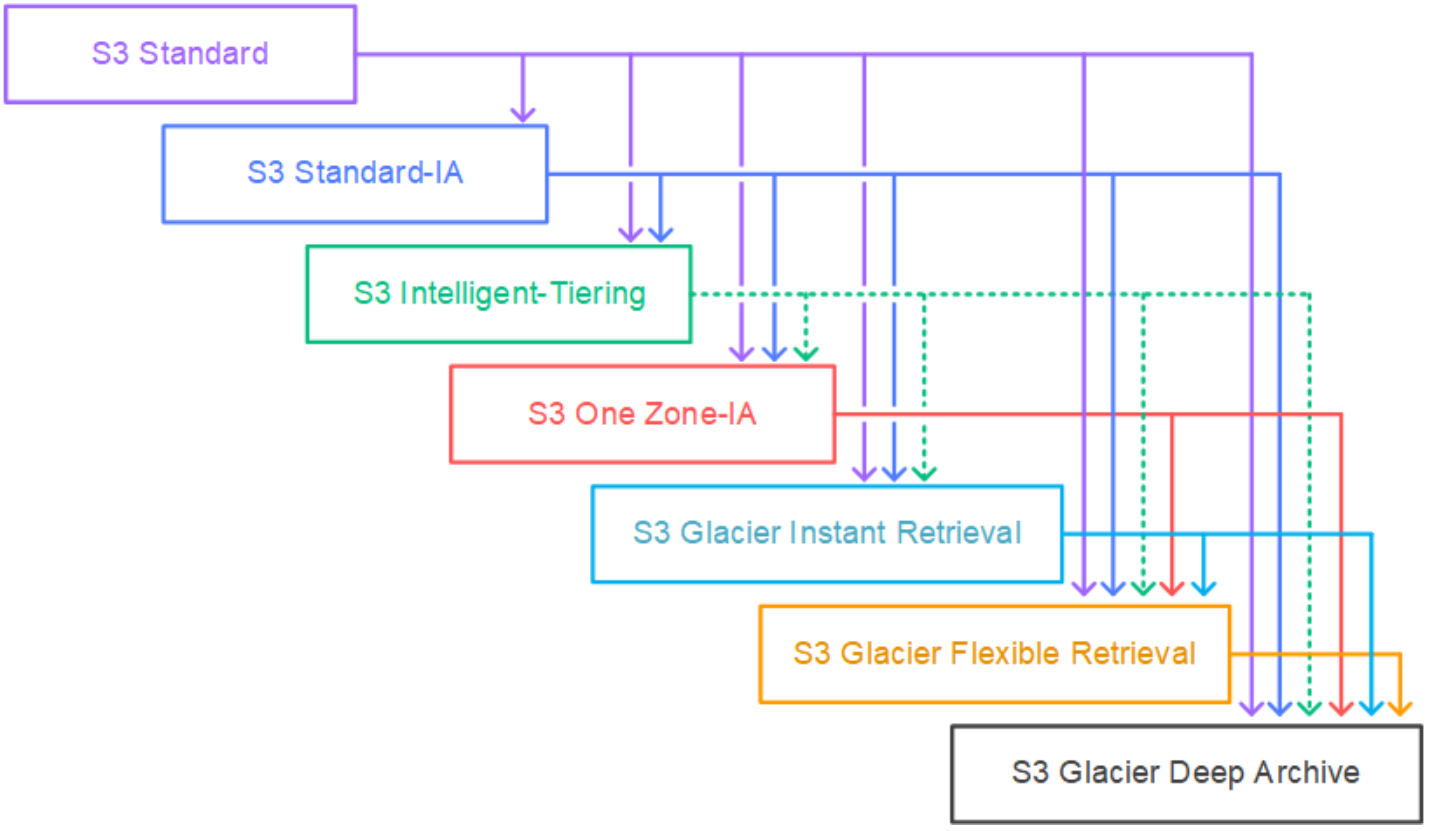
- When you know that objects are infrequently accessed, you might transition them to the S3 Standard-IA storage class.
- You might want to archive objects that you don't need to access in real time to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes.

> **Note**
>
> Encrypted objects remain encrypted throughout the storage class transition process.

Supported transitions

In an S3 Lifecycle configuration, you can define rules to transition objects from one storage class to another to save on storage costs. When you don't know the access patterns of your objects, or if your access patterns are changing over time, you can transition the objects to the S3 Intelligent-Tiering storage class for automatic cost savings. For information about storage classes, see Understanding and managing Amazon S3 storage classes.

Amazon S3 supports a waterfall model for transitioning between storage classes, as shown in the following diagram.



Supported lifecycle transitions

Amazon S3 supports the following lifecycle transitions between storage classes using an S3 Lifecycle configuration.

- The S3 Standard storage class to the S3 Standard-IA, S3 Intelligent-Tiering, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage classes.
- The S3 Standard-IA storage class to the S3 Intelligent-Tiering, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage classes.
- The S3 Intelligent-Tiering storage class can transition to different storage classes depending on the S3 Intelligent-Tiering access tier. The following transitions are possible for each access tier.
  - Frequent Access tier or Infrequent Access tier to S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage classes.
  - Archive Instant Access tier to S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage classes.
  - Archive Access tier to S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive storage classes.
  - Deep Archive Access tier to S3 Glacier Deep Archive storage classes.
- The S3 One Zone-IA storage class to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes.
- The S3 Glacier Instant Retrieval storage class to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes.
- The S3 Glacier Flexible Retrieval storage class to the S3 Glacier Deep Archive storage class.

> **Note**
>
> For versioning enabled or versioning suspended buckets, you can't transition objects with a `Pending` or `Failed` replication status.

Constraints and considerations for transitions

Lifecycle storage class transitions have the following constraints:

Objects smaller than 128 KB will not transition by default to any storage class

Amazon S3 applies a default behavior to S3 Lifecycle configurations that prevents objects smaller than 128 KB from being transitioned to any storage class. We don't recommend transitioning objects less than 128 KB because you are charged a transition request for each object. This means, for smaller objects, the transition costs can outweigh the storage savings. For more information about transition request costs, see **Requests & data retrievals** on the **Storage & requests** tab of the Amazon S3 pricing page.

To allow smaller objects to transition, you can add an object size filter to your Lifecycle transition rules that specifies a custom minimum size (`ObjectSizeGreaterThan`) or maximum size (`ObjectSizeLessThan`). For more information, see Example: Allowing objects smaller than 128 KB to be transitioned.

> **Note**
>
> In September 2024 Amazon S3 updated the default transition behavior for small objects, as follows:
>
> - **Updated default transition behavior** — Starting September 2024, the default behavior prevents objects smaller than 128 KB from being transitioned to any storage class.
> - **Previous default transition behavior** — Before September 2024, the default behavior allowed objects smaller than 128 KB to be transitioned only to the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes.
>
> Configurations created before September 2024 retain the previous transition behavior unless you modify them. That is, if you create, edit, or delete rules, the default transition behavior for your configuration changes to the updated behavior. If your use case requires, you can change the default transition behavior so that objects smaller than 128KB will transition to S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive. To do this, use the optional `x-amz-transition-default-minimum-object-size` header in a PutBucketLifecycleConfiguration request.

You are charged for transitioning objects before their minimum storage duration

Certain storage classes have a minimum object storage duration. If you transition objects out of these storage classes before the minimum duration, you are charged for the remainder of that duration. For more information on which storage classes have minimum storage durations, see Comparing the Amazon S3 storage classes.

You can't create a single Lifecycle rule that transitions objects from one storage class to another before the minimum storage duration period has passed.

For example, S3 Glacier Instant Retrieval has a minimum storage duration of 90 days. You can't specify a lifecycle rule that transitions objects to S3 Glacier Instant Retrieval after 4 days, and

then transitions objects to S3 Glacier Deep Archive after 20 days. In this case the S3 Glacier Deep Archive transition must occur after at least 94 days.

You can specify two rules to accomplish this, but you pay the minimum duration storage charges. For more information about cost considerations, see Amazon S3 pricing.

For more information about creating a S3 Lifecycle, see Setting an S3 Lifecycle configuration on a bucket.

Tag-based filters and eligibility evaluation

S3 Lifecycle evaluates objects against tag-based filters daily. When an object matches a tag-based transition rule, Amazon S3 queues the action for asynchronous processing. At execution time, Amazon S3 re-evaluates the object's current tags. If the triggering tag is no longer present when the action executes, the transition does not proceed.

Because the time between evaluation and execution is not deterministic, removing a tag does not guarantee an immediate cancellation — the action may execute before the tag removal is observed, or the tag removal may take effect before the next evaluation cycle.

To reliably prevent a transition after tagging an object, wait for a Lifecycle Event Notification confirming the action completed before removing the tag. If you need to prevent a transition before it executes, remove the tag rather than disabling the rule (rule policy updates can take up to 15 minutes to propagate).

Transitioning to the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes (object archival)

By using an S3 Lifecycle configuration, you can transition objects to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes for archiving.

Before you archive objects, review the following sections for relevant considerations.

General considerations

The following are the general considerations for you to consider before you archive objects:

- Encrypted objects remain encrypted throughout the storage class transition process.
- Objects that are stored in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes are not available in real time.

Archived objects are Amazon S3 objects, but before you can access an archived object, you must first restore a temporary copy of it. The restored object copy is available only for the duration that you specify in the restore request. After that, Amazon S3 deletes the temporary copy, and the object remains archived in S3 Glacier Flexible Retrieval.

You can restore an object by using the Amazon S3 console or programmatically by using the AWS SDK wrapper libraries or the Amazon S3 REST API in your code. For more information, see Restoring an archived object.

- Objects that are stored in the S3 Glacier Flexible Retrieval storage class can only be transitioned to the S3 Glacier Deep Archive storage class.

  You can use an S3 Lifecycle configuration rule to convert the storage class of an object from S3 Glacier Flexible Retrieval to the S3 Glacier Deep Archive storage class only. If you want to change the storage class of an object that is stored in S3 Glacier Flexible Retrieval to a storage class other than S3 Glacier Deep Archive, you must use the restore operation to make a temporary copy of the object first. Then use the copy operation to overwrite the object specifying S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-IA, or Reduced Redundancy as the storage class.

- The transition of objects to the S3 Glacier Deep Archive storage class can go only one way.

  You cannot use an S3 Lifecycle configuration rule to convert the storage class of an object from S3 Glacier Deep Archive to any other storage class. If you want to change the storage class of an archived object to another storage class, you must use the restore operation to make a temporary copy of the object first. Then use the copy operation to overwrite the object specifying S3 Standard, S3 Intelligent-Tiering, S3 Standard-IA, S3 One Zone-IA, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or Reduced Redundancy Storage as the storage class.

> **Note**
>
> The Copy operation for restored objects isn't supported in the Amazon S3 console for objects in the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. For this type of Copy operation, use the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the REST API.

The objects that are stored in the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes are visible and available only through Amazon S3. They are not available through the separate Amazon Glacier service.

These are Amazon S3 objects, and you can access them only by using the Amazon S3 console or the Amazon S3 API. You cannot access the archived objects through the separate Amazon Glacier console or the Amazon Glacier API.

Cost considerations

If you are planning to archive infrequently accessed data for a period of months or years, the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes can reduce your storage costs. However, to ensure that the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage class is appropriate for you, consider the following:

- **Storage overhead charges** – When you transition objects to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage class, a fixed amount of storage is added to each object to accommodate metadata for managing the object.
  - For each object archived to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, Amazon S3 uses 8 KB of storage for the name of the object and other metadata. Amazon S3 stores this metadata so that you can get a real-time list of your archived objects by using the Amazon S3 API. For more information, see Get Bucket (List Objects). You are charged S3 Standard rates for this additional storage.
  - For each object that is archived to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, Amazon S3 adds 32 KB of storage for index and related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive rates for this additional storage.

  If you are archiving small objects, consider these storage charges. Also consider aggregating many small objects into a smaller number of large objects to reduce overhead costs.
- **Number of days you plan to keep objects archived** – S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive are long-term archival solutions. The minimal storage duration period is 90 days for the S3 Glacier Flexible Retrieval storage class and 180 days for S3 Glacier Deep Archive. Deleting data that is archived to Amazon Glacier doesn't incur charges if the objects you delete are archived for more than the minimal storage duration period. If you delete or overwrite an archived object within the minimal duration period, Amazon S3 charges a prorated early deletion

fee. For information about the early deletion fee, see the "How am I charged for deleting objects from Amazon Glacier that are less than 90 days old?" question on the Amazon S3 FAQ.

- **S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive transition request charges** – Each object that you transition to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage class constitutes one transition request. There is a cost for each such request. If you plan to transition a large number of objects, consider the request costs. If you are archiving a mix of objects that includes small objects, especially those under 128KB, we recommend using the lifecycle object size filter to filter out small objects from your transition to reduce request costs.
- **S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive data restore charges** – S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive are designed for long-term archival of data that you access infrequently. For information about data restoration charges, see the "How much does it cost to retrieve data from Amazon Glacier?" question on the Amazon S3 FAQ. For information about how to restore data from Amazon Glacier, see Restoring an archived object.

> **Note**
>
> S3 Lifecycle transitions objects to S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive asynchronously. There might be a delay between the transition date in the S3 Lifecycle configuration rule and the date of the physical transition. You are charged at the destination storage class rate (S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive) starting from the date the lifecycle rule is satisfied, even if the physical transition has not yet occurred. The minimum storage duration charges (90 days for S3 Glacier Flexible Retrieval, 180 days for S3 Glacier Deep Archive) and the per-object storage overhead (40 KB total, of which 8 KB is charged at S3 Standard rates and 32 KB at the destination Glacier rate) also begin at this point. The only exception is transitions to S3 Intelligent-Tiering, where billing changes occur after the physical transition completes. For more information, see Changes in billing.

The Amazon S3 product detail page provides pricing information and example calculations for archiving Amazon S3 objects. For more information, see the following topics:

- "How is my storage charge calculated for Amazon S3 objects archived to Amazon Glacier?" on the Amazon S3 FAQ.
- "How am I charged for deleting objects from Amazon Glacier that are less than 90 days old?" on the Amazon S3 FAQ.

- "How much does it cost to retrieve data from Amazon Glacier?" on the Amazon S3 FAQ.
- Amazon S3 pricing for storage costs for the different storage classes.

Restoring archived objects

Archived objects aren't accessible in real time. You must first initiate a restore request and then wait until a temporary copy of the object is available for the duration that you specify in the request. After you receive a temporary copy of the restored object, the object's storage class remains S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive. (A HeadObject or GetObject API operation request will return S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive as the storage class.)

> **Note**
>
> When you restore an archive, you are paying for both the archive (S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive rate) and a copy that you restored temporarily (S3 Standard storage rate). For information about pricing, see Amazon S3 pricing.

You can restore an object copy programmatically or by using the Amazon S3 console. Amazon S3 processes only one restore request at a time per object. For more information, see Restoring an archived object.

Expiring objects

You can add expiration actions to an S3 Lifecycle configuration to tell Amazon S3 to delete objects at the end of their lifetime. When an object reaches the end of its lifetime based on its lifecycle configuration, Amazon S3 takes an `Expiration` action based on which S3 Versioning state the bucket is in:

- **Nonversioned bucket** – Amazon S3 queues the object for removal and removes it asynchronously, permanently removing the object.
- **Versioning-enabled bucket** – If the current object version is not a delete marker, Amazon S3 adds a delete marker with a unique version ID. This makes the current version noncurrent, and the delete marker the current version.
- **Versioning-suspended bucket** – Amazon S3 creates a delete marker with null as the version ID. This delete marker replaces any object version with a null version ID in the version hierarchy, which effectively deletes the object.

For a versioned bucket (that is, versioning-enabled or versioning-suspended), there are several considerations that guide how Amazon S3 handles the `Expiration` action. For versioning-enabled or versioning-suspended buckets, the following applies:

- Object expiration applies only to an object's current version (it has no impact on noncurrent object versions).
- Amazon S3 doesn't take any action if there are one or more object versions and the delete marker is the current version.
- If the current object version is the only object version and it is also a delete marker (also referred as an *expired object delete marker*, where all object versions are deleted and you only have a delete marker remaining), Amazon S3 removes the expired object delete marker. You can also use the `Expiration` action to direct Amazon S3 to remove any expired object delete markers. For example, see Removing expired object delete markers in a versioning-enabled bucket.
- You can use the `NoncurrentVersionExpiration` action element to direct Amazon S3 to permanently delete noncurrent versions of objects. These deleted objects can't be recovered. You can base this expiration on a certain number of days since the objects became noncurrent. In addition to the number of days, you can also provide a maximum number of noncurrent versions to retain (between 1 and 100). This value specifies how many newer noncurrent versions must exist before Amazon S3 can perform the associated action on a given version. To specify the maximum number of noncurrent versions, you must also provide a `Filter` element. If you don't specify a `Filter` element, Amazon S3 generates an `InvalidRequest` error when you provide a maximum number of noncurrent versions. For more information about using the `NoncurrentVersionExpiration` action element, see the section called “Elements to describe lifecycle actions”.
- Amazon S3 doesn't take any action on noncurrent versions of objects that have the S3 Object Lock configuration applied.
- For objects with a `Pending` or `Failed` replication status, Amazon S3 doesn't take any action on current or non-current versions of objects.

For more information, see Retaining multiple versions of objects with S3 Versioning.

> **Important**
>
> An object is eligible for only one S3 Lifecycle action per day. When multiple rules match, Amazon S3 applies the least costly action.

Important

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both an S3 Glacier Flexible Retrieval and an S3 Standard-IA (or an S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Examples of overlapping filters and conflicting lifecycle actions.

Existing and new objects

When you add a Lifecycle configuration to a bucket, the configuration rules apply to both existing objects and objects that you add later. For example, if you add a Lifecycle configuration rule today with an expiration action that causes objects with a specific prefix to expire 30 days after creation, Amazon S3 will queue for removal any existing objects that are more than 30 days old and that have the specified prefix.

Important

You can't use a bucket policy to prevent deletions or transitions by an S3 Lifecycle rule. For example, even if your bucket policy denies all actions for all principals, your S3 Lifecycle configuration still functions as normal.

Tag-based filters and eligibility evaluation

S3 Lifecycle evaluates objects against tag-based filters daily. When an object matches a tag-based expiration rule, Amazon S3 queues the action for asynchronous processing. At execution time, Amazon S3 re-evaluates the object's current tags. If the triggering tag is no longer present when the action executes, the expiration does not proceed.

To reliably confirm an expiration occurred before removing the tag, wait for a Lifecycle Event Notification. If you need to prevent an expiration before it executes, remove the tag (tag updates take effect faster than rule policy changes, which can take up to 15 minutes to propagate).

How to find when objects will expire

To find when the current version of an object is scheduled to expire, use the HeadObject or GetObject API operation. These API operations return response headers that provide the date and time at which the current version of the object is no longer cacheable.

> **Note**
>
> - There may be a delay between the expiration date and the date at which Amazon S3 removes an object. You are not charged for expiration or the storage time associated with an object that has expired. For more information, including guidance for deletions that must meet a deadline, see Transition or expiration delay.
> - Before updating, disabling, or deleting Lifecycle rules, use the `LIST` API operations (such as ListObjectsV2, ListObjectVersions, and ListMultipartUploads) or Cataloging and analyzing your data with S3 Inventory to verify that Amazon S3 has transitioned and expired eligible objects based on your use cases.

Minimum storage duration charge

If you create an S3 Lifecycle expiration rule that causes objects that have been in S3 Standard-IA or S3 One Zone-IA storage for less than 30 days to expire, you are charged for 30 days. If you create a Lifecycle expiration rule that causes objects that have been in S3 Glacier Flexible Retrieval storage for less than 90 days to expire, you are charged for 90 days. If you create a Lifecycle expiration rule that causes objects that have been in S3 Glacier Deep Archive storage for less than 180 days to expire, you are charged for 180 days.

For more information, see Amazon S3 pricing.

Setting an S3 Lifecycle configuration on a bucket

You can set an Amazon S3 Lifecycle configuration on a bucket by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API. For information about S3 Lifecycle configuration, see Managing the lifecycle of objects.

> **Note**
>
> To view or edit the lifecycle configuration for a directory bucket, use the AWS CLI, AWS SDKs, or the Amazon S3 REST API. For more information, see Working with S3 Lifecycle for directory buckets.

In your S3 Lifecycle configuration, you use *lifecycle rules* to define actions that you want Amazon S3 to take during an object's lifetime. For example, you can define rules to transition objects to another storage class, archive objects, or expire (delete) objects after a specified period of time.

S3 Lifecycle considerations

Before you set a lifecycle configuration, note the following:

Lifecycle configuration propagation delay

When you add an S3 Lifecycle configuration to a bucket, there is usually some lag before a new or updated Lifecycle configuration is fully propagated to all the Amazon S3 systems. Expect a delay of a few minutes before the configuration fully takes effect. This delay can also occur when you delete an S3 Lifecycle configuration.

Transition or expiration delay

There's a delay between when a lifecycle rule is satisfied and when the action for the rule is completed. For example, suppose that a set of objects is expired by a lifecycle rule on January 1. Even though the expiration rule has been satisfied on January 1, Amazon S3 might not actually delete these objects until days or even weeks later. This delay occurs because S3 Lifecycle queues objects for transitions or expirations asynchronously. S3 Lifecycle doesn't guarantee a completion time for transition or expiration actions. To track progress for your bucket, see How do I monitor the actions taken by my lifecycle rules? When you add or modify a Lifecycle rule, S3 Lifecycle may begin processing eligible objects immediately or with some delay. When S3 Lifecycle creates a delete marker or transitions an object, the timestamp is set to midnight UTC on the day the action occurred, regardless of the actual time the action was taken. A delay doesn't affect your bill. Changes in billing are applied when the object becomes eligible for the lifecycle action, even if the action isn't complete. The one exception is transitions to S3 Intelligent-Tiering, which are billed after the transition completes. For more information, see Changes in billing.

Note

We don't recommend relying on S3 Lifecycle alone for deletions that must complete by a fixed deadline, such as a regulatory or contractual data-retention requirement. Use an Cataloging and analyzing your data with S3 Inventory report with the Lifecycle Expiration Date field to identify objects that are past their expiration date but haven't been removed yet, and delete any objects that must be removed by your deadline.

Note

When a lifecycle rule is created or modified, objects that already meet the eligibility criteria may be processed immediately.

Updating, disabling, or deleting lifecycle rules

When you disable or delete lifecycle rules, Amazon S3 stops scheduling new objects for deletion or transition after a small delay. Any objects that were already scheduled are unscheduled and are not deleted or transitioned.

Note

Before updating, disabling, or deleting lifecycle rules, use the `LIST` API operations (such as ListObjectsV2, ListObjectVersions, and ListMultipartUploads) or Cataloging and analyzing your data with S3 Inventory to verify that Amazon S3 has transitioned and expired eligible objects based on your use cases. If you're experiencing any issues with updating, disabling, or deleting lifecycle rules, see Troubleshooting Amazon S3 Lifecycle issues.

Existing and new objects

When you add a Lifecycle configuration to a bucket, the configuration rules apply to both existing objects and objects that you add later. For example, if you add a Lifecycle configuration rule today with an expiration action that causes objects with a specific prefix to expire 30 days after creation, Amazon S3 will queue for removal any existing objects that are more than 30 days old and that have the specified prefix.

Monitoring the effect of lifecycle rules

To monitor the effect of updates made by active lifecycle rules, see the section called "How do I monitor the actions taken by my lifecycle rules?"

Changes in billing

There might be a lag between when the Lifecycle configuration rules are satisfied and when the action triggered by satisfying the rule is taken. However, changes in billing happen as soon as the Lifecycle configuration rule is satisfied, even if the action isn't yet taken.

For example, after the object expiration time, you aren't charged for storage, even if the object isn't deleted immediately. Likewise, as soon as the object transition time elapses, you're charged S3 Glacier Flexible Retrieval storage rates, even if the object isn't immediately transitioned to the S3 Glacier Flexible Retrieval storage class.

However, lifecycle transitions to the S3 Intelligent-Tiering storage class are the exception. Changes in billing don't happen until after the object has transitioned into the S3 Intelligent-Tiering storage class.

For transitions to S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive, the per-object storage overhead (40 KB per object) and minimum storage duration charges (90 days for S3 Glacier Flexible Retrieval, 180 days for S3 Glacier Deep Archive) also begin when the lifecycle rule is satisfied, regardless of when the physical transition occurs. For more information about Glacier storage overhead, see Cost considerations.

Multiple or conflicting rules

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both an S3 Glacier Flexible Retrieval and an S3 Standard-IA (or an S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Examples of overlapping filters and conflicting lifecycle actions.

How to set an S3 Lifecycle configuration

You can set an Amazon S3 Lifecycle configuration on a general purpose bucket by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the AWS SDKs, or the Amazon S3 REST API.

For information about AWS CloudFormation templates and examples, see Working with AWS CloudFormation templates and AWS::S3::Bucket in the *CloudFormation User Guide*.

Using the S3 console

You can define lifecycle rules for all objects or a subset of objects in a bucket by using a shared prefix (objects names that begin with a common string) or a tag. In your lifecycle rule, you can define actions specific to current and noncurrent object versions. For more information, see the following:

- Managing the lifecycle of objects
- Retaining multiple versions of objects with S3 Versioning

To create a lifecycle rule

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to create a lifecycle rule for.
4. Choose the **Management** tab, and choose **Create lifecycle rule**.
5. In **Lifecycle rule name**, enter a name for your rule.

   The name must be unique within the bucket.
6. Choose the scope of the lifecycle rule:

   - To apply this lifecycle rule to *all objects with a specific prefix or tag*, choose **Limit the scope to specific prefixes or tags**.
     - To limit the scope by prefix, in **Prefix**, enter the prefix.
     - To limit the scope by tag, choose **Add tag**, and enter the tag key and value.

     For more information about object name prefixes, see Naming Amazon S3 objects. For more information about object tags, see Tagging your objects.

- To apply this lifecycle rule to *all objects in the bucket*, choose **This rule applies to *all* objects in the bucket**, and then choose **I acknowledge that this rule applies to all objects in the bucket**.

7. To filter a rule by object size, you can select **Specify minimum object size**, **Specify maximum object size**, or both options.

   - When you're specifying a value for **Minimum object size** or **Maximum object size**, the value must be larger than 0 bytes and up to 50 TB. You can specify this value in bytes, KB, MB, or GB.
   - When you're specifying both values, the maximum object size must be larger than the minimum object size.

   > **Note**
   >
   > The **Minimum object size** and **Maximum object size** filters exclude the specified values. For example, if you set a filter to expire objects that have a **Minimum object size** of 128 KB, objects that are exactly 128 KB don't expire. Instead, the rule applies only to objects that are greater than 128 KB in size.

8. Under **Lifecycle rule actions**, choose the actions that you want your lifecycle rule to perform:

   - Transition *current* versions of objects between storage classes
   - Transition *previous* versions of objects between storage classes
   - Expire *current* versions of objects

   > **Note**
   >
   > For buckets that don't have S3 Versioning enabled, expiring current versions causes Amazon S3 to permanently delete the objects. For more information, see the section called “Lifecycle actions and bucket versioning state”.

   - Permanently delete *previous* versions of objects
   - Delete expired delete markers or incomplete multipart uploads

   Depending on the actions that you choose, different options appear.

9. To transition *current* versions of objects between storage classes, under **Transition current versions of objects between storage classes**, do the following:

   a. In **Storage class transitions**, choose the storage class to transition to. For a list of possible transitions, see the section called "Supported lifecycle transitions". You can choose from the following storage classes:

      - S3 Standard-IA
      - S3 Intelligent-Tiering
      - S3 One Zone-IA
      - S3 Glacier Instant Retrieval
      - S3 Glacier Flexible Retrieval
      - S3 Glacier Deep Archive

   b. In **Days after object creation**, enter the number of days after creation to transition the object.

   For more information about storage classes, see Understanding and managing Amazon S3 storage classes. You can define transitions for current or previous object versions or for both current and previous versions. Versioning enables you to keep multiple versions of an object in one bucket. For more information about versioning, see Using the S3 console.

   > **Important**
   >
   > When you choose the S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or Glacier Deep Archive storage class, your objects remain in Amazon S3. You cannot access them directly through the separate Amazon Glacier service. For more information, see Transitioning objects using Amazon S3 Lifecycle.

10. To transition *noncurrent* versions of objects between storage classes, under **Transition noncurrent versions of objects between storage classes**, do the following:

    a. In **Storage class transitions**, choose the storage class to transition to. For a list of possible transitions, see the section called "Supported lifecycle transitions". You can choose from the following storage classes:

       - S3 Standard-IA
       - S3 Intelligent-Tiering

- S3 One Zone-IA
- S3 Glacier Instant Retrieval
- S3 Glacier Flexible Retrieval
- S3 Glacier Deep Archive

b. In **Days after object becomes noncurrent**, enter the number of days after creation to transition the object.

11. To expire *current* versions of objects, under **Expire current versions of objects**, in **Number of days after object creation**, enter the number of days.

> **Important**
>
> In a nonversioned bucket, the expiration action results in Amazon S3 permanently removing the object. For more information about lifecycle actions, see Elements to describe lifecycle actions.

12. To permanently delete previous versions of objects, under **Permanently delete noncurrent versions of objects**, in **Days after objects become noncurrent**, enter the number of days. You can optionally specify the number of newer versions to retain by entering a value under **Number of newer versions to retain**.
13. Under **Delete expired delete markers or incomplete multipart uploads**, choose **Delete expired object delete markers** and **Delete incomplete multipart uploads**. Then, enter the number of days after the multipart upload initiation that you want to end and clean up incomplete multipart uploads.

    For more information about multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3.
14. Choose **Create rule**.

    If the rule does not contain any errors, Amazon S3 enables it, and you can see it on the **Management** tab under **Lifecycle rules**.

Using the AWS CLI

You can use the following AWS CLI commands to manage S3 Lifecycle configurations:

- `put-bucket-lifecycle-configuration`

- `get-bucket-lifecycle-configuration`
- `delete-bucket-lifecycle`

For instructions on setting up the AWS CLI, see Developing with Amazon S3 using the AWS CLI in the *Amazon S3 API Reference*.

The Amazon S3 Lifecycle configuration is an XML file. But when you're using the AWS CLI, you cannot specify the XML format. You must specify the JSON format instead. The following are example XML lifecycle configurations and the equivalent JSON configurations that you can specify in an AWS CLIcommand.

Consider the following example S3 Lifecycle configuration.

Example 1

Example

XML

```
<LifecycleConfiguration>
    <Rule>
        <ID>ExampleRule</ID>
        <Filter>
           <Prefix>documents/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
</LifecycleConfiguration>
```

JSON

```
{
    "Rules": [
        {
```

```
            "Filter": {
                "Prefix": "documents/"
            },
            "Status": "Enabled",
            "Transitions": [
                {
                    "Days": 365,
                    "StorageClass": "GLACIER"
                }
            ],
            "Expiration": {
                "Days": 3650
            },
            "ID": "ExampleRule"
        }
    ]
}
```

Example 2

Example

XML

```
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
    <Rule>
        <ID>id-1</ID>
        <Expiration>
            <Days>1</Days>
        </Expiration>
        <Filter>
            <And>
                <Prefix>myprefix</Prefix>
                <Tag>
                    <Key>mytagkey1</Key>
                    <Value>mytagvalue1</Value>
                </Tag>
                <Tag>
                    <Key>mytagkey2</Key>
                    <Value>mytagvalue2</Value>
                </Tag>
            </And>
        </Filter>
```

```
        <Status>Enabled</Status>
    </Rule>
</LifecycleConfiguration>
```

JSON

```
{
    "Rules": [
        {
            "ID": "id-1",
            "Filter": {
                "And": {
                    "Prefix": "myprefix",
                    "Tags": [
                        {
                            "Value": "mytagvalue1",
                            "Key": "mytagkey1"
                        },
                        {
                            "Value": "mytagvalue2",
                            "Key": "mytagkey2"
                        }
                    ]
                }
            },
            "Status": "Enabled",
            "Expiration": {
                "Days": 1
            }
        }
    ]
}
```

You can test the `put-bucket-lifecycle-configuration` as follows.

To test the configuration

1. Save the JSON Lifecycle configuration in a file (for example, *`lifecycle.json`*).
2. Run the following AWS CLI command to set the Lifecycle configuration on your bucket. Replace the *`user input placeholders`* with your own information.

```
$ aws s3api put-bucket-lifecycle-configuration  \
--bucket amzn-s3-demo-bucket  \
--lifecycle-configuration file://lifecycle.json
```

3. To verify, retrieve the S3 Lifecycle configuration by using the `get-bucket-lifecycle-configuration` AWS CLI command as follows:

```
$ aws s3api get-bucket-lifecycle-configuration  \
--bucket amzn-s3-demo-bucket
```

> **Important**
>
> The output of `get-bucket-lifecycle-configuration` includes `TransitionDefaultMinimumObjectSize` as a top-level field alongside `Rules`. This field is not part of the `LifecycleConfiguration` JSON schema accepted by `put-bucket-lifecycle-configuration`. If you use the GET output directly as the PUT input, the command fails with a validation error.
> To copy a lifecycle configuration between buckets, you must either remove `TransitionDefaultMinimumObjectSize` from the JSON file or use the `--query '{Rules: Rules}'` parameter with the GET command to extract only the rules. If you need to preserve the `TransitionDefaultMinimumObjectSize` value, pass it separately using the `--transition-default-minimum-object-size` parameter.

4. To delete the S3 Lifecycle configuration, use the `delete-bucket-lifecycle` AWS CLI command as follows:

```
aws s3api delete-bucket-lifecycle \
--bucket amzn-s3-demo-bucket
```

Using the AWS SDKs

Java

You can use the AWS SDK for Java to manage the S3 Lifecycle configuration of a bucket. For more information about managing S3 Lifecycle configuration, see Managing the lifecycle of objects.

> **Note**
>
> When you add an S3 Lifecycle configuration to a bucket, Amazon S3 replaces the bucket's current Lifecycle configuration, if there is one. To update a configuration, you retrieve it, make the desired changes, and then add the revised configuration to the bucket.

To manage lifecycle configuration using the AWS SDK for Java, you can:

- Add a Lifecycle configuration to a bucket.
- Retrieve the Lifecycle configuration and update it by adding another rule.
- Add the modified Lifecycle configuration to the bucket. Amazon S3 replaces the existing configuration.
- Retrieve the configuration again and verify that it has the right number of rules by printing the number of rules.
- Delete the Lifecycle configuration and verify that it has been deleted by attempting to retrieve it again.

For examples of how to set lifecycle configuration on a bucket with the AWS SDK for Java, see Set lifecycle configuration on a bucket in the *Amazon S3 API Reference*.

.NET

You can use the AWS SDK for .NET to manage the S3 Lifecycle configuration on a bucket. For more information about managing Lifecycle configuration, see Managing the lifecycle of objects.

> **Note**
>
> When you add a Lifecycle configuration, Amazon S3 replaces the existing configuration on the specified bucket. To update a configuration, you must first retrieve the Lifecycle configuration, make the changes, and then add the revised Lifecycle configuration to the bucket.

The following example shows how to use the AWS SDK for .NET to add, update, and delete a bucket's Lifecycle configuration. The code example does the following:

- Adds a Lifecycle configuration to a bucket.
- Retrieves the Lifecycle configuration and updates it by adding another rule.
- Adds the modified Lifecycle configuration to the bucket. Amazon S3 replaces the existing Lifecycle configuration.
- Retrieves the configuration again and verifies it by printing the number of rules in the configuration.
- Deletes the Lifecycle configuration and verifies the deletion.

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class LifecycleTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddUpdateDeleteLifecycleConfigAsync().Wait();
        }

        private static async Task AddUpdateDeleteLifecycleConfigAsync()
        {
            try
            {
                var lifeCycleConfiguration = new LifecycleConfiguration()
                {
                    Rules = new List<LifecycleRule>
```

```
                        {
                            new LifecycleRule
                            {
                                 Id = "Archive immediately rule",
                                 Filter = new LifecycleFilter()
                                 {
                                     LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                     {
                                         Prefix = "glacierobjects/"
                                     }
                                 },
                                 Status = LifecycleRuleStatus.Enabled,
                                 Transitions = new List<LifecycleTransition>
                                 {
                                      new LifecycleTransition
                                      {
                                           Days = 0,
                                           StorageClass = S3StorageClass.Glacier
                                      }
                                  },
                            },
                            new LifecycleRule
                            {
                                 Id = "Archive and then delete rule",
                                  Filter = new LifecycleFilter()
                                 {
                                     LifecycleFilterPredicate = new
LifecyclePrefixPredicate()
                                     {
                                         Prefix = "projectdocs/"
                                     }
                                 },
                                 Status = LifecycleRuleStatus.Enabled,
                                 Transitions = new List<LifecycleTransition>
                                 {
                                      new LifecycleTransition
                                      {
                                           Days = 30,
                                           StorageClass =
S3StorageClass.StandardInfrequentAccess
                                      },
                                      new LifecycleTransition
                                      {
```

```
                                    Days = 365,
                                    StorageClass = S3StorageClass.Glacier
                                }
                            },
                            Expiration = new LifecycleRuleExpiration()
                            {
                                Days = 3650
                            }
                        }
                    }
                };

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Retrieve an existing configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

                // Add a new rule.
                lifeCycleConfiguration.Rules.Add(new LifecycleRule
                {
                    Id = "NewRule",
                    Filter = new LifecycleFilter()
                    {
                        LifecycleFilterPredicate = new LifecyclePrefixPredicate()
                        {
                            Prefix = "YearlyDocuments/"
                        }
                    },
                    Expiration = new LifecycleRuleExpiration()
                    {
                        Days = 3650
                    }
                });

                // Add the configuration to the bucket.
                await AddExampleLifecycleConfigAsync(client,
lifeCycleConfiguration);

                // Verify that there are now three rules.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);
                Console.WriteLine("Expected # of rulest=3; found:{0}",
lifeCycleConfiguration.Rules.Count);
```

```
                // Delete the configuration.
                await RemoveLifecycleConfigAsync(client);

                // Retrieve a nonexistent configuration.
                lifeCycleConfiguration = await RetrieveLifecycleConfigAsync(client);

            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered ***. Message:'{0}' when writing
an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
LifecycleConfiguration configuration)
        {

            PutLifecycleConfigurationRequest request = new
PutLifecycleConfigurationRequest
            {
                BucketName = bucketName,
                Configuration = configuration
            };
            var response = await client.PutLifecycleConfigurationAsync(request);
        }

        static async Task<LifecycleConfiguration>
RetrieveLifecycleConfigAsync(IAmazonS3 client)
        {
            GetLifecycleConfigurationRequest request = new
GetLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            var response = await client.GetLifecycleConfigurationAsync(request);
            var configuration = response.Configuration;
            return configuration;
```

```
        }

        static async Task RemoveLifecycleConfigAsync(IAmazonS3 client)
        {
            DeleteLifecycleConfigurationRequest request = new
 DeleteLifecycleConfigurationRequest
            {
                BucketName = bucketName
            };
            await client.DeleteLifecycleConfigurationAsync(request);
        }
    }
}
```

Ruby

You can use the AWS SDK for Ruby to manage an S3 Lifecycle configuration on a bucket by using the class AWS::S3::BucketLifecycleConfiguration. For more information about managing S3 Lifecycle configuration, see Managing the lifecycle of objects.

Using the REST API

The following topics in the *Amazon Simple Storage Service API Reference* describe the REST API operations related to S3 Lifecycle configuration:

- PutBucketLifecycleConfiguration
- GetBucketLifecycleConfiguration
- DeleteBucketLifecycle

Troubleshooting S3 Lifecycle

For common issues that might occur when working with S3 Lifecycle, see the section called "Troubleshooting lifecycle issues".

How S3 Lifecycle interacts with other bucket configurations

In addition to S3 Lifecycle configurations, you can associate other configurations with your bucket. This section explains how S3 Lifecycle configuration relates to other bucket configurations.

S3 Lifecycle and S3 Versioning

You can add S3 Lifecycle configurations to unversioned buckets and versioning-enabled buckets. For more information, see Retaining multiple versions of objects with S3 Versioning.

A versioning-enabled bucket maintains one current object version, and zero or more noncurrent object versions. You can define separate lifecycle rules for current and noncurrent object versions.

For more information, see Lifecycle configuration elements.

Important

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both a S3 Glacier Flexible Retrieval and S3 Standard-IA (or S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Examples of overlapping filters and conflicting lifecycle actions.

S3 Lifecycle and S3 Replication

When you have both S3 Replication and S3 Lifecycle enabled on a bucket, S3 Lifecycle blocks expiration and transition actions on objects with `PENDING` or `FAILED` replication status. This ensures that Lifecycle does not act on objects until they have successfully replicated to their destination bucket.

Objects transition to a `FAILED` replication state for issues such as missing replication role permissions, AWS Key Management Service (AWS KMS) permissions, or bucket permissions. For more information, see Troubleshooting replication.

Objects with `FAILED` replication status will continue to incur storage costs past their Lifecycle expiration or transition eligibility date until the replication issue is resolved. Once you fix the

underlying replication configuration or IAM permissions, new objects will replicate automatically. However, objects that already have `FAILED` replication status will not automatically retry—you must use S3 Batch Replication to replicate them, or delete them using S3 Batch Operations with AWS Lambda if no longer needed. After objects successfully replicate (or are deleted), Lifecycle will resume processing them according to your configured rules. To identify objects with `FAILED` replication status, you can use Amazon CloudWatch metrics (`OperationFailedReplication`) to monitor failure counts and trends at the bucket level, or use Amazon S3 Inventory reports, Amazon S3 API (`HeadObject` or `GetObject`), or Amazon S3 Event Notifications for object-level details.

S3 Lifecycle configuration on MFA-enabled buckets

S3 Lifecycle configuration on multi-factor authentication buckets configured for MFA delete isn't supported. For more information, see Configuring MFA delete.

S3 Lifecycle and logging

Amazon S3 Lifecycle actions aren't captured by AWS CloudTrail object level logging. CloudTrail captures API requests made to external Amazon S3 endpoints, whereas S3 Lifecycle actions are performed by using internal Amazon S3 endpoints.

You can enable Amazon S3 server access logs in an S3 bucket to capture S3 Lifecycle-related actions, such as object transitions to another storage class and object expirations that result in permanent deletion or logical deletion. For more information, see the section called “Logging server access”.

If you have logging enabled on your bucket, Amazon S3 server access logs report the results of the following operations.

Operation log	Description
`S3.EXPIRE.OBJECT`	Amazon S3 permanently deletes the object because of the lifecycle `Expiration` action.
`S3.CREATE.DELETEMARKER`	Amazon S3 logically deletes the current version by adding a delete marker in a versioning-enabled bucket.
`S3.TRANSITION_SIA.OBJECT`	Amazon S3 transitions the object to the S3 Standard-IA storage class.

Operation log	Description
S3.TRANSITION_ZIA.OBJECT	Amazon S3 transitions the object to the S3 One Zone-IA storage class.
S3.TRANSITION_INT.OBJECT	Amazon S3 transitions the object to the S3 Intelligent-Tiering storage class.
S3.TRANSITION_GIR.OBJECT	Amazon S3 initiates the transition of the object to the S3 Glacier Instant Retrieval storage class.
S3.TRANSITION.OBJECT	Amazon S3 initiates the transition of the object to the S3 Glacier Flexible Retrieval storage class.
S3.TRANSITION_GDA.OBJECT	Amazon S3 initiates the transition of the object to the S3 Glacier Deep Archive storage class.
S3.DELETE.UPLOAD	Amazon S3 aborts an incomplete multipart upload.

Note

Amazon S3 server access log records are delivered on a best-effort basis and can't be used for a complete accounting of all Amazon S3 requests.

Configuring S3 Lifecycle event notifications

To receive notice when Amazon S3 deletes an object or transitions it to another Amazon S3 storage class as the result of following an S3 Lifecycle rule, you can set up an Amazon S3 event notification.

You can receive notifications for the following S3 Lifecycle events:

- **Transition events** – By using the `s3:LifecycleTransition` event type, you can receive notification when an object is transitioned from one Amazon S3 storage class to another by an S3 Lifecycle configuration.
- **Expiration (delete) events** – By using the `LifecycleExpiration` event types, you can receive notifications whenever Amazon S3 deletes an object based on your S3 Lifecycle configuration.

  There are two expiration event types:

  - The `s3:LifecycleExpiration:Delete` event type notifies you when an object in an unversioned bucket is deleted. `s3:LifecycleExpiration:Delete` also notifies you when an object version is permanently deleted by an S3 Lifecycle configuration.
  - The `s3:LifecycleExpiration:DeleteMarkerCreated` event type notifies you when S3 Lifecycle creates a delete marker after a current version of an object in a versioned bucket is deleted. S3 Lifecycle sets the delete marker's creation time to 00:00 UTC (midnight) of the current day. This creation time might differ from the event time in the `s3:LifecycleExpiration:DeleteMarkerCreated` notification that S3 sends. For more information, see the section called “Deleting object versions”.

Amazon S3 can publish event notifications to an Amazon Simple Notification Service (Amazon SNS) topic, an Amazon Simple Queue Service (Amazon SQS) queue, or an AWS Lambda function. For more information, see Amazon S3 Event Notifications.

For instructions on how to configure Amazon S3 Event Notifications, see Enabling event notifications by using Amazon SQS, Amazon SNS, and AWS Lambda.

The following is an example of a message that Amazon S3 sends to publish an `s3:LifecycleExpiration:Delete` event. For more information, see the section called “Event message structure”.

```
{
   "Records":[
      {
         "eventVersion":"2.3",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"LifecycleExpiration:Delete",
         "userIdentity":{
            "principalId":"s3.amazonaws.com"
         },
```

```
         "requestParameters":{
            "sourceIPAddress":"s3.amazonaws.com"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"amzn-s3-demo-bucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
               "arn":"arn:aws:s3:::amzn-s3-demo-bucket"
            },
            "object":{
               "key":"expiration/delete",
               "sequencer":"0055AED6DCD90281E5",
            }
         }
      }
   ]
}
```

Messages that Amazon S3 sends to publish an `s3:LifecycleTransition` event also include the following information:

```
"lifecycleEventData":{
    "transitionEventData": {
        "destinationStorageClass": the destination storage class for the object
    }
}
```

Lifecycle configuration elements

A S3 Lifecycle configuration consist of Lifecycle rules that include various elements that describe the actions Amazon S3 takes during an object's lifetime. Each S3 bucket can have one lifecycle configuration assigned to it, which can contain up to 1,000 rules. You specify an Amazon S3

Lifecycle configuration as XML, consisting of one or more Lifecycle rules, where each rule consists of one or more elements.

```
<LifecycleConfiguration>
    <Rule>
         <Element>
    </Rule>
    <Rule>
         <Element>
         <Element>
    </Rule>
</LifecycleConfiguration>
```

Each rule consists of the following:

- Rule metadata that includes a rule ID, and a status that indicates whether the rule is enabled or disabled. If a rule is disabled, Amazon S3 doesn't perform any actions specified in the rule.
- A filter that identifies the objects to which the rule applies. You can specify a filter by using the object size, the object key prefix, one or more object tags, or a combination of filters.
- One or more transition or expiration actions with a date or a time period in the object's lifetime when you want Amazon S3 to perform the specified action.

Topics

- ID element
- Status element
- Filter element
- Elements to describe lifecycle actions
- Adding filters to Lifecycle rules

The following sections describe the XML elements in an S3 Lifecycle configuration. For example configurations, see Examples of S3 Lifecycle configurations.

ID element

Lifecycle configurations are set at the bucket level, with each bucket having its own lifecycle configuration. An S3 Lifecycle configuration can have up to 1,000 rules per bucket. This limit is not

adjustable. The <ID> element uniquely identifies a rule within a bucket's lifecycle configuration. ID length is limited to 255 characters.

Status element

The <Status> element value can be either Enabled or Disabled. If a rule is disabled, Amazon S3 doesn't perform any of the actions defined in the rule.

Filter element

A S3 Lifecycle rule can apply to all or a subset of objects in a bucket based on the <Filter> element that you specify in the rule.

You can filter objects by key prefix, object tags, or a combination of both (in which case Amazon S3 uses a logical AND to combine the filters). For examples and more information about filters see, Adding filters to Lifecycle rules.

- **Specifying a filter by using key prefixes** – This example shows an S3 Lifecycle rule that applies to a subset of objects based on the key name prefix (logs/). For example, the Lifecycle rule applies to the objects logs/mylog.txt, logs/temp1.txt, and logs/test.txt. The rule does not apply to the object example.jpg.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Prefix>logs/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>
    ...
</LifecycleConfiguration>
```

  If you want to apply a lifecycle action to a subset of objects based on different key name prefixes, specify separate rules. In each rule, specify a prefix-based filter. For example, to describe a lifecycle action for objects with the key prefixes projectA/ and projectB/, you specify two rules as follows:

```
<LifecycleConfiguration>
    <Rule>
```

```
        <Filter>
           <Prefix>projectA/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>

    <Rule>
        <Filter>
           <Prefix>projectB/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>
</LifecycleConfiguration>
```

For more information about object keys, see Naming Amazon S3 objects.

- **Specifying a filter based on object tags** – In the following example, the Lifecycle rule specifies a filter based on a tag (*key*) and value (*value*). The rule then applies only to a subset of objects with the specific tag.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Tag>
              <Key>key</Key>
              <Value>value</Value>
           </Tag>
        </Filter>
        transition/expiration actions
        ...
    </Rule>
</LifecycleConfiguration>
```

You can specify a filter based on multiple tags. You must wrap the tags in the <And> element, as shown in the following example. The rule directs Amazon S3 to perform lifecycle actions on objects with two tags (with the specific tag key and value).

```
<LifecycleConfiguration>
    <Rule>
      <Filter>
```

```
          <And>
             <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
             </Tag>
             <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
             </Tag>
              ...
          </And>
      </Filter>
      transition/expiration actions
    </Rule>
</Lifecycle>
```

The Lifecycle rule applies to objects that have both of the tags specified. Amazon S3 performs a logical AND. Note the following:

- Each tag must match *both* the key and value exactly. If you specify only a <Key> element and no <Value> element, the rule will apply only to objects that match the tag key and that do *not* have a value specified.
- The rule applies to a subset of objects that has all the tags specified in the rule. If an object has additional tags specified, the rule will still apply.

> **Note**
>
> When you specify multiple tags in a filter, each tag key must be unique.

- **Specifying a filter based on both the prefix and one or more tags** – In a Lifecycle rule, you can specify a filter based on both the key prefix and one or more tags. Again, you must wrap all of these filter elements in the <And> element, as follows:

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
          <And>
             <Prefix>key-prefix</Prefix>
             <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
```

```
            </Tag>
            <Tag>
               <Key>key2</Key>
               <Value>value2</Value>
            </Tag>
             ...
          </And>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

Amazon S3 combines these filters by using a logical AND. That is, the rule applies to the subset of objects with the specified key prefix and the specified tags. A filter can have only one prefix, and zero or more tags.

- You can specify an **empty filter**, in which case the rule applies to all objects in the bucket.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

- To filter a rule by **object size**, you can specify a minimum size (`ObjectSizeGreaterThan`) or a maximum size (`ObjectSizeLessThan`), or you can specify a range of object sizes.

  Object size values are in bytes. By default, objects smaller than 128 KB will not be transitioned to any storage class, unless you specify a smaller minimum size (`ObjectSizeGreaterThan`) or a maximum size (`ObjectSizeLessThan`). For more information, see Example: Allowing objects smaller than 128 KB to be transitioned.

```
                    <LifecycleConfiguration>
    <Rule>
        <Filter>
            <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
        </Filter>
```

```
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

> **Note**
>
> The `ObjectSizeGreaterThan` and `ObjectSizeLessThan` filters exclude the specified values. For example, if you set objects sized 128 KB to 1024 KB to move from the S3 Standard storage class to the S3 Standard-IA storage class, objects that are exactly 1024 KB and 128 KB won't transition to S3 Standard-IA. Instead, the rule will apply only to objects that are greater than 128 KB and less than 1024 KB in size.

If you're specifying an object size range, the `ObjectSizeGreaterThan` integer must be less than the `ObjectSizeLessThan` value. When using more than one filter, you must wrap the filters in an <And> element. The following example shows how to specify objects in a range between 500 bytes and 64,000 bytes.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <And>
                <Prefix>key-prefix</Prefix>
                <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>64000</ObjectSizeLessThan>
            </And>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

Elements to describe lifecycle actions

You can direct Amazon S3 to perform specific actions in an object's lifetime by specifying one or more of the following predefined actions in an S3 Lifecycle rule. The effect of these actions depends on the versioning state of your bucket.

- **`Transition` action element** – You specify the `Transition` action to transition objects from one storage class to another. For more information about transitioning objects, see Supported transitions. When a specified date or time period in the object's lifetime is reached, Amazon S3 performs the transition.

  For a versioned bucket (versioning-enabled or versioning-suspended bucket), the `Transition` action applies to the current object version. To manage noncurrent versions, Amazon S3 defines the `NoncurrentVersionTransition` action (described later in this topic).

- **`Expiration` action element** – The `Expiration` action expires objects identified in the rule and applies to eligible objects in any of the Amazon S3 storage classes. For more information about storage classes, see Understanding and managing Amazon S3 storage classes. Amazon S3 makes all expired objects unavailable. Whether the objects are permanently removed depends on the versioning state of the bucket.

  - **Nonversioned bucket** – The `Expiration` action results in Amazon S3 permanently removing the object.
  - **Versioned bucket** – For a versioned bucket (that is, versioning-enabled or versioning-suspended), there are several considerations that guide how Amazon S3 handles the `Expiration` action. For versioning-enabled or versioning-suspended buckets, the following applies:
    - The `Expiration` action applies only to the current version (it has no impact on noncurrent object versions).
    - Amazon S3 doesn't take any action if there are one or more object versions and the delete marker is the current version.
    - If the current object version is the only object version and it is also a delete marker (also referred as an *expired object delete marker*, where all object versions are deleted and you only have a delete marker remaining), Amazon S3 removes the expired object delete marker. You can also use the expiration action to direct Amazon S3 to remove any expired object delete markers. For an example, see Removing expired object delete markers in a versioning-enabled bucket.

    For more information, see Retaining multiple versions of objects with S3 Versioning.

    Also consider the following when setting up Amazon S3 to manage expiration:

    - **Versioning-enabled bucket**

If the current object version is not a delete marker, Amazon S3 adds a delete marker with a unique version ID. This makes the current version noncurrent, and the delete marker the current version.

- **Versioning-suspended bucket**

  In a versioning-suspended bucket, the expiration action causes Amazon S3 to create a delete marker with `null` as the version ID. This delete marker replaces any object version with a null version ID in the version hierarchy, which effectively deletes the object.

In addition, Amazon S3 provides the following actions that you can use to manage noncurrent object versions in a versioned bucket (that is, versioning-enabled and versioning-suspended buckets).

- **`NoncurrentVersionTransition` action element** – Use this action to specify when Amazon S3 transitions objects to the specified storage class. You can base this transition on a certain number of days since the objects became noncurrent (`<NoncurrentDays>`). In addition to the number of days, you can also specify the number of noncurrent versions (`<NewerNoncurrentVersions>`) to retain (between 1 and 100). This value determines how many newer noncurrent versions must exist before Amazon S3 can transition a given version. Amazon S3 will transition any additional noncurrent versions beyond the specified number to retain. For the transition to occur, both the `<NoncurrentDays>` **and** the `<NewerNoncurrentVersions>` values must be exceeded.

  To specify the number of noncurrent versions to retain, you must also provide a `<Filter>` element. If you don't specify a `<Filter>` element, Amazon S3 generates an `InvalidRequest` error when you specify the number of noncurrent versions to retain.

  For more information about transitioning objects, see Supported transitions. For details about how Amazon S3 calculates the date when you specify the number of days in the `NoncurrentVersionTransition` action, see Lifecycle rules: Based on an object's age.
- **`NoncurrentVersionExpiration` action element** – Use this action to direct Amazon S3 to permanently delete noncurrent versions of objects. These deleted objects can't be recovered. You can base this expiration on a certain number of days since the objects became noncurrent (`<NoncurrentDays>`). In addition to the number of days, you can also specify the number of noncurrent versions (`<NewerNoncurrentVersions>`) to retain (between 1 and 100). This value specifies how many newer noncurrent versions must exist before Amazon S3 can expire a given

version. Amazon S3 will permanently delete any additional noncurrent versions beyond the specified number to retain. For the deletion to occur, both the `<NoncurrentDays>` **and** the `<NewerNoncurrentVersions>` values must be exceeded.

To specify the number of noncurrent versions to retain, you must also provide a `<Filter>` element. If you don't specify a `<Filter>` element, Amazon S3 generates an `InvalidRequest` error when you specify the number of noncurrent versions to retain.

Delayed removal of noncurrent objects can be helpful when you need to correct any accidental deletes or overwrites. For example, you can configure an expiration rule to delete noncurrent versions five days after they become noncurrent. For example, suppose that on 1/1/2014 at 10:30 AM UTC, you create an object called `photo.gif` (version ID 111111). On 1/2/2014 at 11:30 AM UTC, you accidentally delete `photo.gif` (version ID 111111), which creates a delete marker with a new version ID (such as version ID 4857693). You now have five days to recover the original version of `photo.gif` (version ID 111111) before the deletion is permanent. On 1/8/2014 at 00:00 UTC, the Lifecycle rule for expiration runs and permanently deletes `photo.gif` (version ID 111111), five days after it became a noncurrent version.

For details about how Amazon S3 calculates the date when you specify the number of days in an `NoncurrentVersionExpiration` action, see Lifecycle rules: Based on an object's age.

> **Note**
>
> Object expiration lifecycle configurations don't remove incomplete multipart uploads. To remove incomplete multipart uploads, you must use the `AbortIncompleteMultipartUpload` Lifecycle configuration action that's described later in this section.

In addition to the transition and expiration actions, you can use the following Lifecycle configuration actions to direct Amazon S3 to stop incomplete multipart uploads or to remove expired object delete markers:

- **`AbortIncompleteMultipartUpload` action element** – Use this element to set a maximum time (in days) that you want to allow multipart uploads to remain in progress. If the applicable multipart uploads (determined by the key name `prefix` specified in the Lifecycle rule) aren't successfully completed within the predefined time period, Amazon S3 stops the incomplete multipart uploads. For more information, see Aborting a multipart upload.

> **Note**
>
> You can't specify this lifecycle action in a rule that has a filter that uses object tags.

- **`ExpiredObjectDeleteMarker` action element** – In a versioning-enabled bucket, a delete marker with zero noncurrent versions is referred to as an *expired object delete marker*. You can use this lifecycle action to direct Amazon S3 to remove expired object delete markers. For an example, see Removing expired object delete markers in a versioning-enabled bucket.

> **Note**
>
> You can't specify this lifecycle action in a rule that has a filter that uses object tags.

How Amazon S3 calculates how long an object has been noncurrent

In a versioning-enabled bucket, you can have multiple versions of an object. There is always one current version, and zero or more noncurrent versions. Each time you upload an object, the current version is retained as the noncurrent version and the newly added version, the successor, becomes the current version. To determine the number of days an object is noncurrent, Amazon S3 looks at when its successor was created. Amazon S3 uses the number of days since its successor was created as the number of days an object is noncurrent.

> **Restoring previous versions of an object when using S3 Lifecycle configurations**
>
> As explained in Restoring previous versions, you can use either of the following two methods to retrieve previous versions of an object:
>
> - **Method 1 – Copy a noncurrent version of the object into the same bucket.** The copied object becomes the current version of that object, and all object versions are preserved.
> - **Method 2 – Permanently delete the current version of the object.** When you delete the current object version, you, in effect, turn the noncurrent version into the current version of that object.
>
> When you're using S3 Lifecycle configuration rules with versioning-enabled buckets, we recommend as a best practice that you use Method 1.

S3 Lifecycle operates under an eventually consistent model. A current version that you permanently deleted might not disappear until the changes propagate to all of the Amazon S3 systems. (Therefore, Amazon S3 might be temporarily unaware of this deletion.) In the meantime, the lifecycle rule that you configured to expire noncurrent objects might permanently remove noncurrent objects, including the one that you want to restore. So, copying the old version, as recommended in Method 1, is the safer alternative.

Lifecycle actions and bucket versioning state

Lifecycle rules: Based on an object's age

You can specify a time period, in the number of days from the creation (or modification) of the object, when Amazon S3 can take the specified action.

When you specify the number of days in the `Transition` and `Expiration` actions in an S3 Lifecycle configuration, note the following:

- The value that you specify is the number of days since object creation when the action will occur.
- Amazon S3 calculates the time by adding the number of days specified in the rule to the object creation time and rounding up the resulting time to the next day at midnight UTC. For example, if an object was created on 1/15/2014 at 10:30 AM UTC and you specify 3 days in a transition rule, then the transition date of the object would be calculated as 1/19/2014 00:00 UTC.

Note

Amazon S3 maintains only the last modified date for each object. For example, the Amazon S3 console shows the **Last modified** date in the object's **Properties** pane. When you initially create a new object, this date reflects the date that the object is created. If you replace the object, the date changes accordingly. Therefore, the creation date is synonymous with the **Last modified** date.

When specifying the number of days in the `NoncurrentVersionTransition` and `NoncurrentVersionExpiration` actions in a Lifecycle configuration, note the following:

- The value that you specify is the number of days from when the version of the object becomes noncurrent (that is, when the object is overwritten or deleted) that Amazon S3 will perform the action on the specified object or objects.
- Amazon S3 calculates the time by adding the number of days specified in the rule to the time when the new successor version of the object is created and rounding up the resulting time to the next day at midnight UTC. For example, in your bucket, suppose that you have a current version of an object that was created on 1/1/2014 at 10:30 AM UTC. If the new version of the object that replaces the current version is created on 1/15/2014 at 10:30 AM UTC, and you specify 3 days in a transition rule, the transition date of the object is calculated as 1/19/2014 00:00 UTC.

Lifecycle rules: Based on a specific date

When specifying an action in an S3 Lifecycle rule, you can specify a date when you want Amazon S3 to take the action. When the specific date arrives, Amazon S3 applies the action to all qualified objects (based on the filter criteria).

If you specify an S3 Lifecycle action with a date that is in the past, all qualified objects become immediately eligible for that lifecycle action.

Important

The date-based action is not a one-time action. Amazon S3 continues to apply the date-based action even after the date has passed, as long as the rule status is `Enabled`. For example, suppose that you specify a date-based `Expiration` action to delete all objects (assume that no filter is specified in the rule). On the specified date, Amazon S3 expires all the objects in the bucket. Amazon S3 also continues to expire any new objects that you create in the bucket. To stop the lifecycle action, you must either remove the action from the lifecycle rule, disable the rule, or delete the rule from the lifecycle configuration.

The date value must conform to the ISO 8601 format. The time is always midnight UTC.

> **Note**
>
> You can't create date-based Lifecycle rules by using the Amazon S3 console, but you can view, disable, or delete such rules.

Adding filters to Lifecycle rules

Filters are an optional Lifecyle rule element that you can use to specify which objects that the rule applies to.

The following elements can be used to filter objects:

Key prefix

You can filter objects based on a prefix. If you want to apply a lifecycle action to a subset of objects with different prefixes, create separate rules for each action.

Object tags

You can filter objects based on one or more tags. Each tag must match both the key and value exactly, and, if you specify multiple tags each tag key must be unique. A filter with multiple object tags applies to a subset of objects that has all the tags specified. If an object has additional tags specified, the filter will still apply.

> **Note**
>
> If you specify only a `Key` element and no `Value` element, the rule will apply only to objects that match the tag key and that do not have a value specified.

Minimum or maximum object size

You can filter objects based on size. You can specify a minimum size (`ObjectSizeGreaterThan`) or a maximum size (`ObjectSizeLessThan`), or you can specify a range of object sizes in the same filter. Object size values are in bytes. Maximum filter size is 50 TB. Amazon S3 applies a default minimum object size to lifecycle configuration. For more information, see Example: Allowing objects smaller than 128 KB to be transitioned.

You can combine different filter elements in which case Amazon S3 uses a logical `AND`.

Filter examples

The following examples show how you can use different filter elements:

- **Specifying a filter by using key prefixes** – This example shows an S3 Lifecycle rule that applies to a subset of objects based on the key name prefix (`logs/`). For example, the Lifecycle rule applies to the objects `logs/mylog.txt`, `logs/temp1.txt`, and `logs/test.txt`. The rule does not apply to the object `example.jpg`.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Prefix>logs/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>
    ...
</LifecycleConfiguration>
```

> **Note**
>
> If you have one or more prefixes that start with the same characters, you can include all of those prefixes in your rule by specifying a partial prefix with no trailing slash (/) in the filter. For example, suppose that you have these prefixes:
>
> ```
> sales1999/
>                sales2000/
>                sales2001/
> ```
>
> To include all three prefixes in your rule, specify `sales` as the prefix in your lifecycle rule.

If you want to apply a lifecycle action to a subset of objects based on different key name prefixes, specify separate rules. In each rule, specify a prefix-based filter. For example, to describe a lifecycle action for objects with the key prefixes `projectA/` and `projectB/`, you specify two rules as follows:

```
<LifecycleConfiguration>
    <Rule>
```

```
        <Filter>
           <Prefix>projectA/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>

    <Rule>
        <Filter>
           <Prefix>projectB/</Prefix>
        </Filter>
        transition/expiration actions
         ...
    </Rule>
</LifecycleConfiguration>
```

For more information about object keys, see Naming Amazon S3 objects.

- **Specifying a filter based on object tags** – In the following example, the Lifecycle rule specifies a filter based on a tag (*key*) and value (*value*). The rule then applies only to a subset of objects with the specific tag.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
           <Tag>
              <Key>key</Key>
              <Value>value</Value>
           </Tag>
        </Filter>
        transition/expiration actions
        ...
    </Rule>
</LifecycleConfiguration>
```

You can specify a filter based on multiple tags. You must wrap the tags in the <And> element, as shown in the following example. The rule directs Amazon S3 to perform lifecycle actions on objects with two tags (with the specific tag key and value).

```
<LifecycleConfiguration>
    <Rule>
      <Filter>
```

```
          <And>
             <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
             </Tag>
             <Tag>
                <Key>key2</Key>
                <Value>value2</Value>
             </Tag>
              ...
          </And>
      </Filter>
      transition/expiration actions
    </Rule>
</Lifecycle>
```

The Lifecycle rule applies to objects that have both of the tags specified. Amazon S3 performs a logical AND. Note the following:

- Each tag must match *both* the key and value exactly. If you specify only a <Key> element and no <Value> element, the rule will apply only to objects that match the tag key and that do *not* have a value specified.
- The rule applies to a subset of objects that has all the tags specified in the rule. If an object has additional tags specified, the rule will still apply.

> **Note**
>
> When you specify multiple tags in a filter, each tag key must be unique.

- **Specifying a filter based on both the prefix and one or more tags** – In a Lifecycle rule, you can specify a filter based on both the key prefix and one or more tags. Again, you must wrap all of these filter elements in the <And> element, as follows:

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
          <And>
             <Prefix>key-prefix</Prefix>
             <Tag>
                <Key>key1</Key>
                <Value>value1</Value>
```

```
            </Tag>
            <Tag>
               <Key>key2</Key>
               <Value>value2</Value>
            </Tag>
             ...
          </And>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

Amazon S3 combines these filters by using a logical AND. That is, the rule applies to the subset of objects with the specified key prefix and the specified tags. A filter can have only one prefix, and zero or more tags.

- **Specifying an empty filter** – You can specify an empty filter, in which case the rule applies to all objects in the bucket.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

- **>Specifying an object size filter** – To filter a rule by object size, you can specify a minimum size (ObjectSizeGreaterThan) or a maximum size (ObjectSizeLessThan), or you can specify a range of object sizes.

  Object size values are in bytes. Maximum filter size is 50 TB. Some storage classes have minimum object size limitations. For more information, see Comparing the Amazon S3 storage classes.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
        </Filter>
        <Status>Enabled</Status>
```

```
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

> **Note**
>
> The `ObjectSizeGreaterThan` and `ObjectSizeLessThan` filters exclude the specified values. For example, if you set objects sized 128 KB to 1024 KB to move from the S3 Standard storage class to the S3 Standard-IA storage class, objects that are exactly 1024 KB and 128 KB won't transition to S3 Standard-IA. Instead, the rule will apply only to objects that are greater than 128 KB and less than 1024 KB in size.

If you're specifying an object size range, the `ObjectSizeGreaterThan` integer must be less than the `ObjectSizeLessThan` value. When using more than one filter, you must wrap the filters in an <And> element. The following example shows how to specify objects in a range between 500 bytes and 64,000 bytes.

```
<LifecycleConfiguration>
    <Rule>
        <Filter>
            <And>
                <Prefix>key-prefix</Prefix>
                <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
                <ObjectSizeLessThan>64000</ObjectSizeLessThan>
            </And>
        </Filter>
        <Status>Enabled</Status>
        transition/expiration actions
    </Rule>
</LifecycleConfiguration>
```

How Amazon S3 handles conflicts in lifecycle configurations

Generally, Amazon S3 Lifecycle optimizes for cost. For example, if two expiration policies overlap, the shorter expiration policy is honored so that data is not stored for longer than expected. Likewise, if two transition policies overlap, S3 Lifecycle transitions your objects to the lower-cost storage class.

In both cases, S3 Lifecycle tries to choose the path that is least expensive for you. An exception to this general rule is with the S3 Intelligent-Tiering storage class. S3 Intelligent-Tiering is favored by S3 Lifecycle over any storage class, aside from the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes.

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both an S3 Glacier Flexible Retrieval and an S3 Standard-IA (or an S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

Examples of overlapping filters and conflicting lifecycle actions

You might specify an S3 Lifecycle configuration in which you specify overlapping prefixes, or actions. The following examples show how Amazon S3 resolves potential conflicts.

Example 1: Overlapping prefixes (no conflict)

The following example configuration has two rules that specify overlapping prefixes as follows:

- The first rule specifies an empty filter, indicating all objects in the bucket.
- The second rule specifies a key name prefix (`logs/`), indicating only a subset of objects.

Rule 1 requests Amazon S3 to delete all objects one year after creation. Rule 2 requests Amazon S3 to transition a subset of objects to the S3 Standard-IA storage class 30 days after creation.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
```

```
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA</StorageClass>
      <Days>30</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

Since there is no conflict in this case, Amazon S3 will transition the objects with the `logs/` prefix to the S3 Standard-IA storage class 30 days after creation. When any object reaches one year after creation, it will be deleted.

Example 2: Conflicting lifecycle actions

In this example configuration, there are two rules that direct Amazon S3 to perform two different actions on the same set of objects at the same time in the objects' lifetime:

- Both rules specify the same key name prefix, so both rules apply to the same set of objects.
- Both rules specify the same 365 days after object creation when the rules apply.
- One rule directs Amazon S3 to transition objects to the S3 Standard-IA storage class and another rule wants Amazon S3 to expire the objects at the same time.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
```

```
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA</StorageClass>
      <Days>365</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

In this case, because you want objects to expire (to be removed), there is no point in changing the storage class, so Amazon S3 chooses the expiration action on these objects.

Example 3: Overlapping prefixes resulting in conflicting lifecycle actions

In this example, the configuration has two rules, which specify overlapping prefixes as follows:

- Rule 1 specifies an empty prefix (indicating all objects).
- Rule 2 specifies a key name prefix (`logs/`) that identifies a subset of all objects.

For the subset of objects with the `logs/` key name prefix, S3 Lifecycle actions in both rules apply. One rule directs Amazon S3 to transition objects 10 days after creation, and another rule directs Amazon S3 to transition objects 365 days after creation.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>STANDARD_IA</StorageClass>
      <Days>10</Days>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
```

```
    <Transition>
      <StorageClass>STANDARD_IA</StorageClass>
      <Days>365</Days>
    </Transition>
   </Rule>
</LifecycleConfiguration>
```

In this case, Amazon S3 chooses to transition them 10 days after creation.

Example 4: Tag-based filtering and resulting conflicting lifecycle actions

Suppose that you have the following S3 Lifecycle configuration that has two rules, each specifying a tag filter:

- Rule 1 specifies a tag-based filter (`tag1/value1`). This rule directs Amazon S3 to transition objects to the S3 Glacier Flexible Retrieval storage class 365 days after creation.
- Rule 2 specifies a tag-based filter (`tag2/value2`). This rule directs Amazon S3 to expire objects 14 days after creation.

The S3 Lifecycle configuration is shown in following example.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Rule 1</ID>
    <Filter>
      <Tag>
         <Key>tag1</Key>
         <Value>value1</Value>
      </Tag>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <StorageClass>GLACIER</StorageClass>
      <Days>365</Days>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule 2</ID>
    <Filter>
      <Tag>
         <Key>tag2</Key>
         <Value>value2</Value>
```

```
      </Tag>
    </Filter>
    <Status>Enabled</Status>
    <Expiration>
      <Days>14</Days>
    </Expiration>
   </Rule>
</LifecycleConfiguration>
```

If an object has both tags, then Amazon S3 has to decide which rule to follow. In this case, Amazon S3 expires the object 14 days after creation. The object is removed, and therefore the transition action does not apply.

Examples of S3 Lifecycle configurations

This section provides examples of S3 Lifecycle configuration. Each example shows how you can specify the XML in each of the example scenarios.

Topics

- Archiving all objects within one day after creation
- Disabling Lifecycle rules temporarily
- Tiering down the storage class over an object's lifetime
- Specifying multiple rules
- Specifying a lifecycle rule for a versioning-enabled bucket
- Removing expired object delete markers in a versioning-enabled bucket
- Lifecycle configuration to abort multipart uploads
- Expiring noncurrent objects that have no data
- Example: Allowing objects smaller than 128 KB to be transitioned

Archiving all objects within one day after creation

Each S3 Lifecycle rule includes a filter that you can use to identify a subset of objects in your bucket to which the S3 Lifecycle rule applies. The following S3 Lifecycle configurations show examples of how you can specify a filter.

- In this S3 Lifecycle configuration rule, the filter specifies a key prefix (`tax/`). Therefore, the rule applies to objects with the key name prefix `tax/`, such as `tax/doc1.txt` and `tax/doc2.txt`.

  The rule specifies two actions that direct Amazon S3 to do the following:

  - Transition objects to the S3 Glacier Flexible Retrieval storage class 365 days (one year) after creation.
  - Delete objects (the `Expiration` action) 3,650 days (10 years) after creation.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition and Expiration Rule</ID>
    <Filter>
       <Prefix>tax/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

  Instead of specifying the object age in terms of days after creation, you can specify a date for each action. However, you can't use both `Date` and `Days` in the same rule.

- If you want the S3 Lifecycle rule to apply to all objects in the bucket, specify an empty prefix. In the following configuration, the rule specifies a `Transition` action that directs Amazon S3 to transition objects to the S3 Glacier Flexible Retrieval storage class 0 days after creation. This rule means that the objects are eligible for archival to S3 Glacier Flexible Retrieval at midnight UTC following creation. For more information about lifecycle constraints, see Constraints and considerations for transitions.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Archive all object same-day upon creation</ID>
    <Filter>
      <Prefix></Prefix>
    </Filter>
```

```
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

- You can specify zero or one key name prefix and zero or more object tags in a filter. The following example code applies the S3 Lifecycle rule to a subset of objects with the `tax/` key prefix and to objects that have two tags with specific key and value. When you specify more than one filter, you must include the <And> element as shown (Amazon S3 applies a logical AND to combine the specified filter conditions).

```
...
<Filter>
   <And>
      <Prefix>tax/</Prefix>
      <Tag>
         <Key>key1</Key>
         <Value>value1</Value>
      </Tag>
      <Tag>
         <Key>key2</Key>
         <Value>value2</Value>
      </Tag>
    </And>
</Filter>
...
```

- You can filter objects based only on tags. For example, the following S3 Lifecycle rule applies to objects that have the two specified tags (it does not specify any prefix).

```
...
<Filter>
   <And>
      <Tag>
         <Key>key1</Key>
         <Value>value1</Value>
      </Tag>
      <Tag>
         <Key>key2</Key>
```

```
          <Value>value2</Value>
      </Tag>
    </And>
</Filter>
...
```

Important

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both an S3 Glacier Flexible Retrieval and S3 Standard-IA (or S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Examples of overlapping filters and conflicting lifecycle actions.

Disabling Lifecycle rules temporarily

You can temporarily disable an S3 Lifecycle rule using the `status` element. This can be useful if you want to test new rules or troubleshoot issues with your configuration, without overwriting your existing rules. The following S3 Lifecycle configuration specifies two rules:

- Rule 1 directs Amazon S3 to transition objects with the `logs/` prefix to the S3 Glacier Flexible Retrieval storage class soon after creation.
- Rule 2 directs Amazon S3 to transition objects with the `documents/` prefix to the S3 Glacier Flexible Retrieval storage class soon after creation.

In the configuration, Rule 1 is enabled and Rule 2 is disabled. Amazon S3 ignores the disabled rule.

```
<LifecycleConfiguration>
```

```
  <Rule>
    <ID>Rule1</ID>
    <Filter>
      <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>Rule2</ID>
    <Filter>
      <Prefix>documents/</Prefix>
    </Filter>
    <Status>Disabled</Status>
    <Transition>
      <Days>0</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

Tiering down the storage class over an object's lifetime

In this example, you use S3 Lifecycle configuration to tier down the storage class of objects over their lifetime. Tiering down can help reduce storage costs. For more information about pricing, see Amazon S3 pricing.

The following S3 Lifecycle configuration specifies a rule that applies to objects with the key name prefix `logs/`. The rule specifies the following actions:

- Two transition actions:
  - Transition objects to the S3 Standard-IA storage class 30 days after creation.
  - Transition objects to the S3 Glacier Flexible Retrieval storage class 90 days after creation.
- One expiration action that directs Amazon S3 to delete objects a year after creation.

```
<LifecycleConfiguration>
  <Rule>
```

```
    <ID>example-id</ID>
    <Filter>
       <Prefix>logs/</Prefix>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Transition>
      <Days>90</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>365</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Note

You can use one rule to describe all S3 Lifecycle actions if all actions apply to the same set of objects (identified by the filter). Otherwise, you can add multiple rules with each specifying a different filter.

Important

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both an S3 Glacier Flexible Retrieval and S3 Standard-IA (or S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Examples of overlapping filters and conflicting lifecycle actions.

Specifying multiple rules

You can specify multiple rules if you want different S3 Lifecycle actions of different objects. The following S3 Lifecycle configuration has two rules:

- Rule 1 applies to objects with the key name prefix `classA/`. It directs Amazon S3 to transition objects to the S3 Glacier Flexible Retrieval storage class one year after creation and expire these objects 10 years after creation.
- Rule 2 applies to objects with key name prefix `classB/`. It directs Amazon S3 to transition objects to the S3 Standard-IA storage class 90 days after creation and delete them one year after creation.

```
<LifecycleConfiguration>
    <Rule>
        <ID>ClassADocRule</ID>
        <Filter>
           <Prefix>classA/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>365</Days>
           <StorageClass>GLACIER</StorageClass>
        </Transition>
        <Expiration>
             <Days>3650</Days>
        </Expiration>
    </Rule>
    <Rule>
        <ID>ClassBDocRule</ID>
        <Filter>
            <Prefix>classB/</Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <Expiration>
             <Days>365</Days>
        </Expiration>
```

```
    </Rule>
</LifecycleConfiguration>
```

Important

When you have multiple rules in an S3 Lifecycle configuration, an object can become eligible for multiple S3 Lifecycle actions on the same day. In such cases, Amazon S3 follows these general rules:

- Permanent deletion takes precedence over transition.
- Transition takes precedence over creation of delete markers.
- When an object is eligible for both an S3 Glacier Flexible Retrieval and S3 Standard-IA (or S3 One Zone-IA) transition, Amazon S3 chooses the S3 Glacier Flexible Retrieval transition.

For examples, see Examples of overlapping filters and conflicting lifecycle actions.

Specifying a lifecycle rule for a versioning-enabled bucket

Suppose that you have a versioning-enabled bucket, which means that for each object, you have a current version and zero or more noncurrent versions. (For more information about S3 Versioning, see Retaining multiple versions of objects with S3 Versioning.)

In the following example, you want to maintain one year's worth of history, and retain 5 noncurrent versions. S3 Lifecycle configurations support keeping 1 to 100 versions of any object. Be aware that more than 5 newer noncurrent versions must exist before Amazon S3 can expire a given version. Amazon S3 will permanently delete any additional noncurrent versions beyond the specified number to retain. For the deletion to occur, both the `NoncurrentDays` and the `NewerNoncurrentVersions` values must be exceeded.

To save storage costs, you want to move noncurrent versions to S3 Glacier Flexible Retrieval 30 days after they become noncurrent (assuming that these noncurrent objects are cold data for which you don't need real-time access). In addition, you expect the frequency of access of the current versions to diminish 90 days after creation, so you might choose to move these objects to the S3 Standard-IA storage class.

```
<LifecycleConfiguration>
```

```
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
           <Prefix></Prefix>
        </Filter>
        <Status>Enabled</Status>
        <Transition>
           <Days>90</Days>
           <StorageClass>STANDARD_IA</StorageClass>
        </Transition>
        <NoncurrentVersionTransition>
            <NoncurrentDays>30</NoncurrentDays>
            <StorageClass>GLACIER</StorageClass>
        </NoncurrentVersionTransition>
       <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>5</NewerNoncurrentVersions>
            <NoncurrentDays>365</NoncurrentDays>
       </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

Removing expired object delete markers in a versioning-enabled bucket

A versioning-enabled bucket has one current version and zero or more noncurrent versions for each object. When you delete an object, note the following:

- If you don't specify a version ID in your delete request, Amazon S3 adds a delete marker instead of deleting the object. The current object version becomes noncurrent, and the delete marker becomes the current version.
- If you specify a version ID in your delete request, Amazon S3 deletes the object version permanently (a delete marker isn't created).
- A delete marker with zero noncurrent versions is referred to as an *expired object delete marker*.

This example shows a scenario that can create expired object delete markers in your bucket, and how you can use S3 Lifecycle configuration to direct Amazon S3 to remove the expired object delete markers.

Suppose that you write an S3 Lifecycle configuration that uses the `NoncurrentVersionExpiration` action to remove noncurrent versions 30 days after they become noncurrent and to retain 10 noncurrent versions, as shown in the following example. Be

aware that more than 10 newer noncurrent versions must exist before Amazon S3 can expire a given version. Amazon S3 will permanently delete any additional noncurrent versions beyond the specified number to retain. For the deletion to occur, both the `NoncurrentDays` and the `NewerNoncurrentVersions` values must be exceeded.

```
<LifecycleConfiguration>
    <Rule>
        ...
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

The `NoncurrentVersionExpiration` action doesn't apply to the current object versions. It removes only the noncurrent versions.

For current object versions, you have the following options to manage their lifetime, depending on whether the current object versions follow a well-defined lifecycle:

- **The current object versions follow a well-defined lifecycle.**

  In this case, you can use an S3 Lifecycle configuration with the `Expiration` action to direct Amazon S3 to remove the current versions, as shown in the following example.

  ```
  <LifecycleConfiguration>
      <Rule>
          ...
          <Expiration>
             <Days>60</Days>
          </Expiration>
          <NoncurrentVersionExpiration>
              <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
              <NoncurrentDays>30</NoncurrentDays>
          </NoncurrentVersionExpiration>
      </Rule>
  </LifecycleConfiguration>
  ```

  In this example, Amazon S3 removes current versions 60 days after they're created by adding a delete marker for each of the current object versions. This process makes the current version

noncurrent, and the delete marker becomes the current version. For more information, see Retaining multiple versions of objects with S3 Versioning.

> **Note**
>
> You can't specify both a `Days` and an `ExpiredObjectDeleteMarker` tag on the same rule. When you specify the `Days` tag, Amazon S3 automatically performs `ExpiredObjectDeleteMarker` cleanup when the delete markers are old enough to satisfy the age criteria. To clean up delete markers as soon as they become the only version, create a separate rule with only the `ExpiredObjectDeleteMarker` tag.

The `NoncurrentVersionExpiration` action in the same S3 Lifecycle configuration removes noncurrent objects 30 days after they become noncurrent. Thus, in this example, all object versions are permanently removed 90 days after object creation. Be aware that in this example, more than 10 newer noncurrent versions must exist before Amazon S3 can expire a given version. Amazon S3 will permanently delete any additional noncurrent versions beyond the specified number to retain. For the deletion to occur, both the `NoncurrentDays` and the `NewerNoncurrentVersions` values must be exceeded.

Although expired object delete markers are created during this process, Amazon S3 detects and removes the expired object delete markers for you.

- **The current object versions don't have a well-defined lifecycle.**

  In this case, you might remove the objects manually when you don't need them, creating a delete marker with one or more noncurrent versions. If your S3 Lifecycle configuration with the `NoncurrentVersionExpiration` action removes all the noncurrent versions, you now have expired object delete markers.

  Specifically for this scenario, S3 Lifecycle configuration provides an `Expiration` action that you can use to remove the expired object delete markers.

```
<LifecycleConfiguration>
    <Rule>
       <ID>Rule 1</ID>
        <Filter>
          <Prefix>logs/</Prefix>
        </Filter>
```

```
        <Status>Enabled</Status>
        <Expiration>
           <ExpiredObjectDeleteMarker>true</ExpiredObjectDeleteMarker>
        </Expiration>
        <NoncurrentVersionExpiration>
            <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
            <NoncurrentDays>30</NoncurrentDays>
        </NoncurrentVersionExpiration>
    </Rule>
</LifecycleConfiguration>
```

By setting the `ExpiredObjectDeleteMarker` element to `true` in the `Expiration` action, you direct Amazon S3 to remove the expired object delete markers.

> **Note**
>
> When you use the `ExpiredObjectDeleteMarker` S3 Lifecycle action, the rule cannot specify a tag-based filter.

Lifecycle configuration to abort multipart uploads

You can use the Amazon S3 multipart upload REST API operations to upload large objects in parts. For more information about multipart uploads, see Uploading and copying objects using multipart upload in Amazon S3.

By using an S3 Lifecycle configuration, you can direct Amazon S3 to stop incomplete multipart uploads (identified by the key name prefix specified in the rule) if they aren't completed within a specified number of days after initiation. When Amazon S3 aborts a multipart upload, it deletes all the parts associated with the multipart upload. This process helps control your storage costs by ensuring that you don't have incomplete multipart uploads with parts that are stored in Amazon S3.

> **Note**
>
> When you use the `AbortIncompleteMultipartUpload` S3 Lifecycle action, the rule cannot specify a tag-based filter.

The following is an example S3 Lifecycle configuration that specifies a rule with the `AbortIncompleteMultipartUpload` action. This action directs Amazon S3 to stop incomplete multipart uploads seven days after initiation.

```
<LifecycleConfiguration>
    <Rule>
        <ID>sample-rule</ID>
        <Filter>
           <Prefix>SomeKeyPrefix/</Prefix>
        </Filter>
        <Status>rule-status</Status>
        <AbortIncompleteMultipartUpload>
          <DaysAfterInitiation>7</DaysAfterInitiation>
        </AbortIncompleteMultipartUpload>
    </Rule>
</LifecycleConfiguration>
```

Expiring noncurrent objects that have no data

You can create rules that transition objects based only on their size. You can specify a minimum size (`ObjectSizeGreaterThan`) or a maximum size (`ObjectSizeLessThan`), or you can specify a range of object sizes in bytes. When using more than one filter, such as a prefix and size rule, you must wrap the filters in an <And> element.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Transition with a prefix and based on size</ID>
    <Filter>
       <And>
          <Prefix>tax/</Prefix>
          <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
       </And>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

If you're specifying a range by using both the `ObjectSizeGreaterThan` and `ObjectSizeLessThan` elements, the maximum object size must be larger than the minimum object size. When using more than one filter, you must wrap the filters in an <And> element. The following example shows how to specify objects in a range between 500 bytes and 64,000 bytes. When you're specifying a range, the `ObjectSizeGreaterThan` and `ObjectSizeLessThan` filters exclude the specified values. For more information, see the section called “Filter element”.

```
<LifecycleConfiguration>
    <Rule>
        ...
          <And>
             <ObjectSizeGreaterThan>500</ObjectSizeGreaterThan>
             <ObjectSizeLessThan>64000</ObjectSizeLessThan>
          </And>
    </Rule>
</LifecycleConfiguration>
```

You can also create rules to specifically expire noncurrent objects that have no data, including noncurrent delete marker objects created in a versioning-enabled bucket. The following example uses the `NoncurrentVersionExpiration` action to remove noncurrent versions 30 days after they become noncurrent and to retain 10 noncurrent versions. This example also uses the `ObjectSizeLessThan` element to filter only objects with no data.

Be aware that more than 10 newer noncurrent versions must exist before Amazon S3 can expire a given version. Amazon S3 will permanently delete any additional noncurrent versions beyond the specified number to retain. For the deletion to occur, both the `NoncurrentDays` and the `NewerNoncurrentVersions` values must be exceeded.

```
<LifecycleConfiguration>
  <Rule>
    <ID>Expire noncurrent with size less than 1 byte</ID>
    <Filter>
       <ObjectSizeLessThan>1</ObjectSizeLessThan>
    </Filter>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
       <NewerNoncurrentVersions>10</NewerNoncurrentVersions>
       <NoncurrentDays>30</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
```

```
</LifecycleConfiguration>
```

Example: Allowing objects smaller than 128 KB to be transitioned

Amazon S3 applies a default behavior to your Lifecycle configuration that prevents objects smaller than 128 KB from being transitioned to any storage class. You can allow smaller objects to transition by adding a minimum size (`ObjectSizeGreaterThan`) or a maximum size (`ObjectSizeLessThan`) filter that specifies a smaller size to the configuration. The following example allows any object smaller than 128 KB to transition to the S3 Glacier Instant Retrieval storage class:

```
<LifecycleConfiguration>
  <Rule>
    <ID>Allow small object transitions</ID>
    <Filter>
          <ObjectSizeGreaterThan>1</ObjectSizeGreaterThan>
    </Filter>
    <Status>Enabled</Status>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER_IR</StorageClass>
    </Transition>
  </Rule>
</LifecycleConfiguration>
```

Note

In September 2024, Amazon S3 updated the default transition behavior for small objects, as follows:

- **Updated default transition behavior** — Starting September 2024, the default behavior prevents objects smaller than 128 KB from being transitioned to any storage class.
- **Previous default transition behavior** — Before September 2024, the default behavior allowed objects smaller than 128 KB to be transitioned only to the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes.

Configurations created before September 2024 retain the previous transition behavior unless you modify them. That is, if you create, edit, or delete rules, the default transition behavior for your configuration changes to the updated behavior. If your use case requires,

you can change the default transition behavior so that objects smaller than 128KB will transition to S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive. To do this, use the optional `x-amz-transition-object-size-minimum-default` header in a PutBucketLifecycleConfiguration request.

The following example shows how to use the `x-amz-transition-object-size-minimum-default` header in a PutBucketLifecycleConfiguration request to apply the `varies_by_storage_class` default transition behavior to an S3 Lifecycle configuration. This behavior allows object smaller than 128 KB to transition to the S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage classes. By default, all other storage classes will prevent transitions smaller than 128 KB. You can still use custom filters to change the minimum transition size for any storage class. Custom filters always take precedence over the default transition behavior:

```
HTTP/1.1 200
x-amz-transition-object-size-minimum-default: varies_by_storage_class
<?xml version="1.0" encoding="UTF-8"?>
...
```

Troubleshooting Amazon S3 Lifecycle issues

The following information can help you troubleshoot common issues with Amazon S3 Lifecycle rules.

Topics

- I ran a list operation on my bucket and saw objects that I thought were expired or transitioned by a lifecycle rule.
- How do I monitor the actions taken by my lifecycle rules?
- My S3 object count still increases, even after setting up lifecycle rules on a versioning-enabled bucket.
- How do I empty my S3 bucket by using lifecycle rules?
- My Amazon S3 bill increased after transitioning objects to a lower-cost storage class.
- I've updated my bucket policy, but my S3 objects are still being deleted by expired lifecycle rules.
- Can I recover S3 objects that are expired by S3 Lifecycle rules?
- Why are my expiration and transition lifecycle actions not occurring?

- How can I exclude a prefix from my lifecycle rule?
- How can I include multiple prefixes in my lifecycle rule?

I ran a list operation on my bucket and saw objects that I thought were expired or transitioned by a lifecycle rule.

S3 Lifecycle object transitions and object expirations are asynchronous operations. Therefore, there might be a delay between the time that the objects are eligible for expiration or transition and the time that they are actually transitioned or expired. Changes in billing are applied as soon as the lifecycle rule is satisfied, even if the action isn't complete. The exception to this behavior is if you have a lifecycle rule set to transition to the S3 Intelligent-Tiering storage class. In that case, billing changes don't occur until the object has transitioned to S3 Intelligent-Tiering. For transitions to S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive, billing at the destination storage class rate, the per-object storage overhead (40 KB), and the minimum storage duration timer all begin when the lifecycle rule is satisfied, even though the physical transition occurs asynchronously. For more information about changes in billing, see Setting lifecycle configuration on a bucket.

> **Note**
>
> Amazon S3 doesn't transition objects that are smaller than 128 KB from the S3 Standard or S3 Standard-IA storage class to the S3 Intelligent-Tiering, S3 Standard-IA, or S3 One Zone-IA storage class.

How do I monitor the actions taken by my lifecycle rules?

To monitor actions taken by lifecycle rules, you can use the following features:

- **S3 Event Notifications** – You can set up S3 Event Notifications so that you're notified of any S3 Lifecycle expiration or transition events.
- **S3 server access logs** – You can enable server access logs for your S3 buckets to capture S3 Lifecycle actions, such as object transitions to another storage class or object expirations. For more information, see Lifecycle and logging.
- **S3 Metadata journal tables** – A journal table records object-level changes in your bucket in near real time, including expirations and transitions made by S3 Lifecycle (the `requester` is `s3.amazonaws.com`). You can query the table to see which objects S3 Lifecycle has removed

and when. The journal table records changes only after you create the metadata configuration. For more information, see S3 Metadata journal tables schema.

- **S3 Inventory** – An S3 Inventory report that includes the Lifecycle Expiration Date field shows the expiration date that your lifecycle rules set for each object. Use it to confirm that your rules apply to the objects you expect, and to find objects that are past their expiration date but haven't been removed yet. For more information, see Cataloging and analyzing your data with S3 Inventory.

Because S3 Lifecycle runs asynchronously, use these features to track progress for your bucket. Event notifications, server access logs, and journal tables show actions as S3 Lifecycle completes them. S3 Inventory shows what remains. For more information, see Transition or expiration delay.

To view the changes in your storage caused by lifecycle actions on a daily basis, we recommend using S3 Storage Lens dashboards instead of using Amazon CloudWatch metrics. In your Storage Lens dashboard, you can view the following metrics, which monitor the object count or size:

- **Current version bytes**
- **Current version object count**
- **Noncurrent version bytes**
- **Noncurrent version object count**
- **Delete marker object count**
- **Delete marker storage bytes**
- **Incomplete multipart upload bytes**
- **Incomplete multipart upload object count**

My S3 object count still increases, even after setting up lifecycle rules on a versioning-enabled bucket.

In a versioning-enabled bucket, when an object is expired, the object isn't completely deleted from the bucket. Instead, a delete marker is created as the newest version of the object. Delete markers are still counted as objects. Therefore, if a lifecycle rule is created to expire only the current versions, then the object count in the S3 bucket actually increases instead of going down.

For example, let's say an S3 bucket is versioning-enabled with 100 objects, and a lifecycle rule is set to expire current versions of the object after 7 days. After the seventh day, the object count

increases to 200 because 100 delete markers are created in addition to the 100 original objects, which are now the noncurrent versions. For more information about S3 Lifecycle configuration rule actions for versioning-enabled buckets, see Setting lifecycle configuration on a bucket.

To permanently remove objects, add an additional lifecycle configuration to delete the previous versions of the objects, expired delete markers, and incomplete multipart uploads. For instructions on how to create new lifecycle rules, see Setting lifecycle configuration on a bucket.

Note

- Amazon S3 rounds the transition or expiration date of an object to midnight UTC the next day.

  When evaluating objects for lifecycle actions, Amazon S3 uses the object creation time in UTC. For example, consider a nonversioned bucket with a lifecycle rule that's configured to expire objects after one day. Suppose that an object was created on January 1 at 17:05 Pacific Daylight Time (PDT), which corresponds to January 2 at 00:05 UTC. The object becomes one day old at 00:05 UTC on January 3, which makes it eligible for expiration when S3 Lifecycle evaluates objects at 00:00 UTC on January 4.

  Because Amazon S3 lifecycle actions occur asynchronously, there might be some delay between the date specified in the lifecycle rule and the actual physical transition of the object. For more information, see Transition or expiration delay.

  For more information, see Lifecycle rules: Based on an object's age.
- For S3 objects that are protected by Object Lock, current versions are not permanently deleted. Instead, a delete marker is added to the objects, making them noncurrent. Noncurrent versions are then preserved and are not permanently expired.

How do I empty my S3 bucket by using lifecycle rules?

S3 Lifecycle rules are an effective tool to empty an S3 bucket with millions of objects. To delete a large number of objects from your S3 bucket, make sure to use these two pairs of lifecycle rules:

- **Expire current versions of objects** and **Permanently delete previous versions of objects**
- **Delete expired delete markers** and **Delete incomplete multipart uploads**

For steps on how to create a lifecycle configuration rule, see Setting lifecycle configuration on a bucket.

> **Note**
>
> For S3 objects that are protected by Object Lock, current versions are not permanently deleted. Instead, a delete marker is added to the objects, making them noncurrent. Noncurrent versions are then preserved and are not permanently expired.

My Amazon S3 bill increased after transitioning objects to a lower-cost storage class.

There are several reasons that your bill might increase after transitioning objects to a lower-cost storage class:

- S3 Glacier overhead charges for small objects

  For each object that is transitioned to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive, a total overhead of 40 KB is associated with this storage update. As part of the 40 KB overhead, 8 KB is used to store metadata and the name of the object. This 8 KB is charged according to S3 Standard rates. The remaining 32 KB is used for indexing and related metadata. This 32 KB is charged according to S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive pricing.

  Therefore, if you're storing many smaller sized objects, we don't recommend using lifecycle transitions. Instead, to reduce any overhead charges, consider aggregating many smaller objects into a smaller number of large objects before storing them in Amazon S3. For more information about cost considerations, see Transitioning to the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes (object archival).
- Minimum storage charges

  Some S3 storage classes have minimum storage-duration requirements. Objects that are deleted, overwritten, or transitioned from those classes before the minimum duration is satisfied are charged a prorated early transition or deletion fee. These minimum storage-duration requirements are as follows:
  - S3 Standard-IA and S3 One Zone-IA – 30 days
  - S3 Glacier Flexible Retrieval and S3 Glacier Instant Retrieval – 90 days
  - S3 Glacier Deep Archive – 180 days

For more information about these requirements, see the *Constraints* section of Transitioning objects using S3 Lifecycle. For general S3 pricing information, see Amazon S3 pricing and the AWS Pricing Calculator.

- Lifecycle transition costs

  Each time an object is transitioned to a different storage class by a lifecycle rule, Amazon S3 counts that transition as one transition request. The costs for these transition requests are in addition to the costs of these storage classes. If you plan to transition a large number of objects, consider the request costs when transitioning to a lower tier. For more information, see Amazon S3 pricing.

I've updated my bucket policy, but my S3 objects are still being deleted by expired lifecycle rules.

`Deny` statements in a bucket policy don't prevent the expiration of the objects defined in a lifecycle rule. Lifecycle actions (such as transitions or expirations) don't use the S3 `DeleteObject` operation. Instead, S3 Lifecycle actions are performed by using internal S3 endpoints. (For more information, see Lifecycle and logging.)

To prevent your lifecycle rule from taking any action, you must edit, delete, or disable the rule.

Can I recover S3 objects that are expired by S3 Lifecycle rules?

The only way to recover objects that are expired by S3 Lifecycle is through versioning, which must be in place before the objects become eligible for expiration. You cannot undo the expiration operations that are performed by lifecycle rules. If objects are permanently deleted by the S3 Lifecycle rules that are in place, you cannot recover these objects. To enable versioning on a bucket, see the section called "Retaining multiple versions of objects".

If you have applied versioning to the bucket and the noncurrent versions of the objects are still intact, you can restore previous versions of the expired objects. For more information about the behavior of S3 Lifecycle rule actions and versioning states, see the *Lifecycle actions and bucket versioning state* table in Elements to describe lifecycle actions.

> **Note**
>
> If the S3 bucket is protected by AWS Backup or S3 Replication, you might also be able to use these features to recover your expired objects.

Why are my expiration and transition lifecycle actions not occurring?

For a versioning-enabled or versioning-suspended bucket, the following considerations guide how Amazon S3 handles the Expiration action:

- Object expiration applies only to an object's current version (it has no impact on noncurrent object versions).
- Amazon S3 doesn't take any action if there are one or more object versions and the delete marker is the current version.
- Amazon S3 doesn't take any action on noncurrent versions of objects that have S3 Object Lock applied.
- For objects with a `PENDING` or `FAILED` replication status, Amazon S3 doesn't take any action on current or noncurrent versions of objects.

Lifecycle storage class transitions have the following constraints:

- By default, objects smaller than 128 KB won't transition to any storage class.
- For versioning enabled or versioning suspended buckets, objects with a `PENDING` or `FAILED` replication status can't be transitioned.

How can I exclude a prefix from my lifecycle rule?

S3 Lifecycle doesn't support excluding prefixes in your rules. Instead, use tags to tag all of the objects that you want to include in the rule. For more information about using tags in your lifecycle rules, see the section called “Archiving all objects within one day after creation”.

How can I include multiple prefixes in my lifecycle rule?

S3 Lifecycle doesn't support including multiple prefixes in your rules. Instead, use tags to tag all of the objects that you want to include in the rule. For more information about using tags in your lifecycle rules, see the section called “Archiving all objects within one day after creation”.

However, if you have one or more prefixes that start with the same characters, you can include all of those prefixes in your rule by specifying a partial prefix with no trailing slash (/) in the filter. For example, suppose that you have these prefixes:

```
sales1999/
sales2000/
sales2001/
```

To include all three prefixes in your rule, specify `<Prefix>sales</Prefix>` in your lifecycle rule.

Logging and monitoring in Amazon S3

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon S3 and your AWS solutions. We recommend collecting monitoring data from all of the parts of your AWS solution so that you can more easily debug a multipoint failure if one occurs. Before you start monitoring Amazon S3, create a monitoring plan that includes answers to the following questions:

- What are your monitoring goals?
- What resources will you monitor?
- How often will you monitor these resources?
- What monitoring tools will you use?
- Who will perform the monitoring tasks?
- Who should be notified when something goes wrong?

For more information about logging and monitoring in Amazon S3, see the following topics.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Monitoring is an important part of maintaining the reliability, availability, and performance of Amazon S3 and your AWS solutions. You should collect monitoring data from all of the parts of your AWS solution so that you can more easily debug a multi-point failure if one occurs. AWS provides several tools for monitoring your Amazon S3 resources and responding to potential incidents.

Amazon CloudWatch Alarms

Using Amazon CloudWatch alarms, you watch a single metric over a time period that you specify. If the metric exceeds a given threshold, a notification is sent to an Amazon SNS topic or AWS Auto Scaling policy. CloudWatch alarms do not invoke actions because they are in a particular state. Rather the state must have changed and been maintained for a specified number of periods. For more information, see Monitoring metrics with Amazon CloudWatch.

AWS CloudTrail Logs

CloudTrail provides a record of actions taken by a user, role, or an AWS service in Amazon S3. Using the information collected by CloudTrail, you can determine the request that was made to Amazon S3, the IP address from which the request was made, who made the request, when it was made, and additional details. For more information, see Logging Amazon S3 API calls using AWS CloudTrail.

Amazon GuardDuty

Amazon GuardDuty is a threat detection service that continuously monitors your accounts, containers, workloads, and the data within your AWS environment to identify potential threats or security risks to your S3 buckets. GuardDuty also provides rich context about the threats that it detects. GuardDuty monitors AWS CloudTrail management logs for threats and surfaces security relevant information. For example, GuardDuty will include factors of an API request, such as the user that made the request, the location the request was made from, and the specific API requested, that could be unusual in your environment. GuardDuty S3 Protection monitors the S3 data events collected by CloudTrail and identifies potentially anomalous and malicious behavior in all the S3 buckets in your environment.

Amazon S3 Access Logs

Server access logs provide detailed records about requests that are made to a bucket. Server access logs are useful for many applications. For example, access log information can be useful in security and access audits. You can deliver logs to Amazon CloudWatch Logs for structured querying and cross-account aggregation, or to an Amazon S3 general purpose bucket in text format at no charge. For more information, see Logging requests with server access logging.

AWS Trusted Advisor

Trusted Advisor draws upon best practices learned from serving hundreds of thousands of AWS customers. Trusted Advisor inspects your AWS environment and then makes recommendations when opportunities exist to save money, improve system availability and performance, or help close security gaps. All AWS customers have access to five Trusted Advisor checks. Customers with a Business or Enterprise support plan can view all Trusted Advisor checks.

Trusted Advisor has the following Amazon S3-related checks:

- Logging configuration of Amazon S3 buckets.
- Security checks for Amazon S3 buckets that have open access permissions.
- Fault tolerance checks for Amazon S3 buckets that don't have versioning enabled, or have versioning suspended.

For more information, see AWS Trusted Advisor in the *Support User Guide*.

Amazon S3 Storage Lens

Amazon S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. You can use S3 Storage Lens metrics to generate summary insights, such as finding out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use S3 Storage Lens metrics to identify cost-optimization opportunities, implement data-protection and security best practices, and improve the performance of application workloads.

S3 Storage Lens aggregates your metrics and displays the information in the Account snapshot section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. For more information, see Understanding Amazon S3 Storage Lens.

Amazon S3 Inventory

Amazon S3 Inventory generates a list of objects and metadata that you can use to query and manage your objects. You can use this inventory report to generate granular data such as object size, last modified date, encryption status and other fields. Those reports are available daily or weekly to automatically give the latest list.

For example, you can use Amazon S3 Inventory to audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs. You can also use Amazon S3 Inventory to simplify and speed up business workflows and big data jobs, which provides a scheduled alternative to the Amazon S3 synchronous `List` API operations. Amazon S3 Inventory doesn't use the `List` API operations to audit your objects and does not affect the request rate of your bucket. For more information, see Cataloging and analyzing your data with S3 Inventory.

Amazon S3 Event Notifications

With the Amazon S3 Event Notifications feature, you receive notifications when certain events happen in your S3 bucket. To enable notifications, add a notification configuration that identifies the events that you want Amazon S3 to publish. For more information, see Amazon S3 Event Notifications.

Amazon S3 and AWS X-Ray

AWS X-Ray integrates with Amazon S3 to trace upstream requests to update your application's S3 buckets. If a service traces requests by using the X-Ray SDK, Amazon S3 can send the tracing headers to downstream event subscribers such as Λ, Amazon SQS, and Amazon SNS. X-Ray enables trace messages for Amazon S3 event notifications. You can use the X-Ray trace map to view the connections between Amazon S3 and other services that your application uses. For more information, see Amazon S3 and X-Ray.

The following security best practices also address logging and monitoring:

- Identify and audit all your Amazon S3 buckets
- Implement monitoring using Amazon Web Services monitoring tools
- Enable AWS Config
- Enable Amazon S3 server access logging
- Use CloudTrail
- Monitor Amazon Web Services security advisories

Topics

- Monitoring tools
- Logging options for Amazon S3
- Logging Amazon S3 API calls using AWS CloudTrail
- Logging requests with server access logging
- Monitoring metrics with Amazon CloudWatch
- Amazon S3 Event Notifications
- Monitoring your storage activity and usage with Amazon S3 Storage Lens
- Cataloging and analyzing your data with S3 Inventory

Monitoring tools

AWS provides various tools that you can use to monitor Amazon S3. You can configure some of these tools to do the monitoring for you, while some of the tools require manual intervention. We recommend that you automate monitoring tasks as much as possible.

Automated monitoring tools

You can use the following automated monitoring tools to watch Amazon S3 and report when something is wrong:

- **Amazon CloudWatch Alarms** – Watch a single metric over a time period that you specify, and perform one or more actions based on the value of the metric relative to a given threshold over a number of time periods. The action is a notification sent to an Amazon Simple Notification Service (Amazon SNS) topic or Amazon EC2 Auto Scaling policy. CloudWatch alarms do not invoke actions simply because they are in a particular state. The state must have changed and been maintained for a specified number of periods. For more information, see Monitoring metrics with Amazon CloudWatch.
- **AWS CloudTrail Log Monitoring** – Share log files between accounts, monitor CloudTrail log files in real time by sending them to CloudWatch Logs, write log processing applications in Java, and validate that your log files have not changed after delivery by CloudTrail. For more information, see Logging Amazon S3 API calls using AWS CloudTrail.

Manual monitoring tools

Another important part of monitoring Amazon S3 involves manually monitoring those items that the CloudWatch alarms don't cover. The Amazon S3, CloudWatch, Trusted Advisor, and other AWS Management Console dashboards provide an at-a-glance view of the state of your AWS environment. You might want to enable *server access logging*, which tracks requests for access to your bucket. Each access log record provides details about a single access request, such as the requester, bucket name, request time, request action, response status, and error code, if any. For more information, see Logging requests with server access logging.

- The Amazon S3 dashboard shows the following:
  - Your buckets and the objects and properties they contain
- The CloudWatch home page shows the following:
  - Current alarms and status
  - Graphs of alarms and resources
  - Service health status

  In addition, you can use CloudWatch to do the following:
  - Create customized dashboards to monitor the services you care about.

- Graph metric data to troubleshoot issues and discover trends.
- Search and browse all your AWS resource metrics.
- Create and edit alarms to be notified of problems.

- AWS Trusted Advisor can help you monitor your AWS resources to improve performance, reliability, security, and cost effectiveness. Four Trusted Advisor checks are available to all users; more than 50 checks are available to users with a Business or Enterprise support plan. For more information, see AWS Trusted Advisor.

  Trusted Advisor has these checks that relate to Amazon S3:

  - Checks of the logging configuration of Amazon S3 buckets.
  - Security checks for Amazon S3 buckets that have open access permissions.
  - Fault tolerance checks for Amazon S3 buckets that do not have versioning enabled, or have versioning suspended.

Logging options for Amazon S3

You can record the actions that are taken by users, roles, or AWS services on Amazon S3 resources and maintain log records for auditing and compliance purposes. To do this, you can use server-access logging, AWS CloudTrail logging, or a combination of both. We recommend that you use CloudTrail for logging bucket-level and object-level actions for your Amazon S3 resources. For more information about each option, see the following sections:

- Logging requests with server access logging
- Logging Amazon S3 API calls using AWS CloudTrail

Amazon S3 offers two delivery paths for server access logs: direct delivery to an Amazon S3 general purpose bucket and delivery to Amazon CloudWatch Logs. Some capabilities in the following table apply to server access logs only through the Amazon CloudWatch Logs delivery path, as noted.

The following table lists the key properties of CloudTrail logs and Amazon S3 server-access logs. To make sure that CloudTrail meets your security requirements, review the table and notes.

Log properties	AWS CloudTrail	Amazon S3 server logs
Can be forwarded to other systems (Amazon CloudWatch Logs, Amazon CloudWatch Events)	Yes	Yes
Deliver logs to more than one destination (for example, send the same logs to two different buckets)	Yes	Yes[3]
Turn on logs for a subset of objects (prefix)	Yes	No
Cross-account log delivery (target and source bucket owned by different accounts)	Yes	Yes[3]
Integrity validation of log file by using digital signature or hashing	Yes	No
Default or choice of encryption for log files	Yes	Yes[3]
Object operations (by using Amazon S3 APIs)	Yes	Yes
Bucket operations (by using Amazon S3 APIs)	Yes	Yes
Searchable UI for logs	Yes	Yes[3]
Fields for Object Lock parameters, Amazon S3 Select properties for log records	Yes	No

Log properties	AWS CloudTrail	Amazon S3 server logs
Fields for `Object Size`, `Total Time`, `Turn-Around Time`, and `HTTP Referer` for log records	Yes	Yes
Lifecycle transitions, expirations, restores	No	Yes
Logging of keys in a batch delete operation	Yes	Yes
Authentication failures[1]	No	Yes
Accounts where logs get delivered	Bucket owner[2], and requester	Bucket owner, and other accounts with Amazon CloudWatch Logs delivery
Performance and Cost	**AWS CloudTrail**	**Amazon S3 Server Logs**
Price	Management events (first delivery) are free; data events incur a fee, in addition to storage of logs	No cost for Amazon S3 bucket delivery in addition to storage of logs; CloudWatch vended logs rates for Amazon CloudWatch Logs delivery[3].
Speed of log delivery	Data events every 5 minutes; management events every 15 minutes	Within a few hours
Log format	JSON	Space-delimited text records for delivery to an Amazon S3 bucket. Structured JSON for delivery to Amazon CloudWatch Logs[3].

Notes

1. CloudTrail does not deliver logs for requests that fail authentication (in which the provided credentials are not valid) or that fail due to redirection (error code `301 Moved Permanently`). However, it does include logs for requests in which authorization fails (`AccessDenied`) and requests that are made by anonymous users.
2. The S3 bucket owner receives CloudTrail logs when the account does not have full access to the object in the request. For more information, see Amazon S3 object-level actions in cross-account scenarios.
3. In addition to direct delivery to an Amazon S3 bucket, you can deliver Amazon S3 server access logs to Amazon CloudWatch Logs. With this delivery path, you gain additional capabilities, including forwarding to other destinations, delivery to multiple destinations, cross-account and cross-Region aggregation, AWS KMS encryption, and interactive querying with CloudWatch Logs Insights. You can also deliver these logs to Amazon S3 Tables in Apache Iceberg format for SQL analytics. For more information, see Logging requests with server access logging.
4. S3 does not support delivery of CloudTrail logs or server access logs to the requester or the bucket owner for VPC endpoint requests when the VPC endpoint policy denies them or for requests that fail before the VPC policy is evaluated.

Logging Amazon S3 API calls using AWS CloudTrail

AWS CloudTrail is a service that provides a record of actions taken by a user, role, or an AWS service. CloudTrail captures all API calls for Amazon S3 as events. The calls captured include calls from the Amazon S3 console and code calls to the Amazon S3 API operations. Using the information collected by CloudTrail, you can determine the request that was made to Amazon S3, the IP address from which the request was made, when it was made, and additional details.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root user or user credentials.
- Whether the request was made on behalf of an IAM Identity Center user.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

CloudTrail is active in your AWS account when you create the account and you automatically have access to the CloudTrail **Event history**. The CloudTrail **Event history** provides a viewable, searchable, downloadable, and immutable record of the past 90 days of recorded management events in an AWS Region. For more information, see Working with CloudTrail Event history in the *AWS CloudTrail User Guide*. There are no CloudTrail charges for viewing the **Event history**.

For an ongoing record of events in your AWS account past 90 days, create a trail or a CloudTrail Lake event data store.

CloudTrail trails

A *trail* enables CloudTrail to deliver log files to an Amazon S3 bucket. All trails created using the AWS Management Console are multi-Region. You can create a single-Region or a multi-Region trail by using the AWS CLI. Creating a multi-Region trail is recommended because you capture activity in all AWS Regions in your account. If you create a single-Region trail, you can view only the events logged in the trail's AWS Region. For more information about trails, see Creating a trail for your AWS account and Creating a trail for an organization in the *AWS CloudTrail User Guide*.

You can deliver one copy of your ongoing management events to your Amazon S3 bucket at no charge from CloudTrail by creating a trail, however, there are Amazon S3 storage charges. For more information about CloudTrail pricing, see AWS CloudTrail Pricing. For information about Amazon S3 pricing, see Amazon S3 Pricing.

CloudTrail Lake event data stores

CloudTrail Lake lets you run SQL-based queries on your events. CloudTrail Lake converts existing events in row-based JSON format to Apache ORC format. ORC is a columnar storage format that is optimized for fast retrieval of data. Events are aggregated into *event data stores*, which are immutable collections of events based on criteria that you select by applying advanced event selectors. The selectors that you apply to an event data store control which events persist and are available for you to query. For more information about CloudTrail Lake, see Working with AWS CloudTrail Lake in the *AWS CloudTrail User Guide*.

CloudTrail Lake event data stores and queries incur costs. When you create an event data store, you choose the pricing option you want to use for the event data store. The pricing option determines the cost for ingesting and storing events, and the default and maximum retention period for the event data store. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

You can store your log files in your bucket for as long as you want, but you can also define Amazon S3 Lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted by using Amazon S3 server-side encryption (SSE).

Using CloudTrail logs with Amazon S3 server access logs and CloudWatch Logs

AWS CloudTrail logs provide a record of actions taken by a user, role, or an AWS service in Amazon S3, while Amazon S3 server access logs provide detailed records for the requests that are made to an S3 bucket. For more information about how the different logs work, and their properties, performance, and costs, see the section called “Logging options”.

You can use AWS CloudTrail logs together with server access logs for Amazon S3. CloudTrail logs provide you with detailed API tracking for Amazon S3 bucket-level and object-level operations. Server access logs for Amazon S3 provide you with visibility into object-level operations on your data in Amazon S3. For more information about server access logs, see Logging requests with server access logging.

You can also use CloudTrail logs together with Amazon CloudWatch for Amazon S3. CloudTrail integration with CloudWatch Logs delivers S3 bucket-level API activity captured by CloudTrail to a CloudWatch log stream in the CloudWatch log group that you specify. You can create CloudWatch alarms for monitoring specific API activity and receive email notifications when the specific API activity occurs. For more information about CloudWatch alarms for monitoring specific API activity, see the AWS CloudTrail User Guide. For more information about using CloudWatch with Amazon S3, see Monitoring metrics with Amazon CloudWatch.

> **Note**
>
> S3 does not support delivery of CloudTrail logs to the requester or the bucket owner for VPC endpoint requests when the VPC endpoint policy denies them.

CloudTrail tracking with Amazon S3 SOAP API calls

CloudTrail tracks Amazon S3 SOAP API calls. Amazon S3 SOAP support over HTTP is deprecated, but it is still available over HTTPS. For more information about Amazon S3 SOAP support, see Appendix: SOAP API in the *Amazon S3 API Reference*.

Important

Newer Amazon S3 features are not supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.

The following table shows Amazon S3 SOAP actions tracked by CloudTrail logging.

SOAP API name	API event name used in CloudTrail log
ListAllMyBuckets	`ListBuckets`
CreateBucket	`CreateBucket`
DeleteBucket	`DeleteBucket`
GetBucketAccessControlPolicy	`GetBucketAcl`
SetBucketAccessControlPolicy	`PutBucketAcl`
GetBucketLoggingStatus	`GetBucketLogging`
SetBucketLoggingStatus	`PutBucketLogging`

For more information about CloudTrail and Amazon S3, see the following topics:

Topics

- Amazon S3 CloudTrail events
- CloudTrail log file entries for Amazon S3 and S3 on Outposts
- Enabling CloudTrail event logging for S3 buckets and objects
- Identifying Amazon S3 requests using CloudTrail

Amazon S3 CloudTrail events

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

This section provides information about the events that S3 logs to CloudTrail.

Amazon S3 data events in CloudTrail

Data events provide information about the resource operations performed on or in a resource (for example, reading or writing to an Amazon S3 object). These are also known as data plane operations. Data events are often high-volume activities. By default, CloudTrail doesn't log data events. The CloudTrail **Event history** doesn't record data events.

Additional charges apply for data events. For more information about CloudTrail pricing, see AWS CloudTrail Pricing.

You can log data events for the Amazon S3 resource types by using the CloudTrail console, AWS CLI, or CloudTrail API operations. For more information about how to log data events, see Logging data events with the AWS Management Console and Logging data events with the AWS Command Line Interface in the *AWS CloudTrail User Guide*.

The following table lists the Amazon S3 resource types for which you can log data events. The **Data event type (console)** column shows the value to choose from the **Data event type** list on the CloudTrail console. The **resources.type value** column shows the `resources.type` value, which you would specify when configuring advanced event selectors using the AWS CLI or CloudTrail APIs. The **Data APIs logged to CloudTrail** column shows the API calls logged to CloudTrail for the resource type.

Data event type (console)	resources.type value	Data APIs logged to CloudTrail
S3	`AWS::S3::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • DeleteObjects • GetObject • GetObjectAcl • GetObjectAttributes • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • GetObjectTorrent • HeadObject • HeadBucket • ListObjectVersions • ListObjects • ListParts • PutObject • PutObjectAcl • PutObjectLegalHold • PutObjectRetention • PutObjectTagging • PutObjectAnnotation • GetObjectAnnotation • ListObjectAnnotations • DeleteObjectAnnotation

Data event type (console)	resources.type value	Data APIs logged to CloudTrail
		• RestoreObject • SelectObjectContent • UpdateObjectEncryption • UploadPart • UploadPartCopy
S3 Express One Zone	`AWS::S3Express::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CreateSession • CopyObject • CreateMultipartUpload • DeleteObject • DeleteObjects • GetObject • GetObjectAttributes • HeadBucket • HeadObject • ListObjectsV2 • ListParts • PutObject • UploadPart • UploadPartCopy

Data event type (console)	resources.type value	Data APIs logged to CloudTrail
S3 Access Point	`AWS::S3::Access Point`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject (same-region copies only) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetObject • GetObjectAcl • GetObjectAttributes • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • HeadBucket • HeadObject • ListObjects • ListObjectsV2 • ListObjectVersions • ListParts • Presign • PutObject • PutObjectLegalHold • PutObjectRetention • PutObjectAcl • PutObjectTagging • RestoreObject • UploadPart

Data event type (console)	resources.type value	Data APIs logged to CloudTrail
		• UploadPartCopy (same-region copies only)
S3 Object Lambda	`AWS::S3ObjectLambda::AccessPoint`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject (same-region copies only) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetObject • GetObjectAcl • GetObjectLegalHold • GetObjectRetention • GetObjectTagging • HeadObject • ListObjects • ListObjectVersions • ListParts • PutObject • PutObjectLegalHold • PutObjectRetention • PutObjectAcl • PutObjectTagging • RestoreObject • UploadPart • WriteGetObjectResponse

Data event type (console)	resources.type value	Data APIs logged to CloudTrail
S3 Outposts	`AWS::S3Outposts::Object`	• AbortMultipartUpload • CompleteMultipartUpload • CopyObject (same-region copies only) • CreateMultipartUpload • DeleteObject • DeleteObjectTagging • GetObject • GetObjectTagging • HeadObject • ListObjects • ListObjectsV2 • ListParts • PutObject • PutObjectTagging • UploadPart • UploadPartCopy

You can configure advanced event selectors to filter on the `eventName`, `readOnly`, and `resources.ARN` fields to log only those events that are important to you. For more information about these fields, see AdvancedFieldSelector in the *AWS CloudTrail API Reference*.

Amazon S3 management events in CloudTrail

Amazon S3 logs all control plane operations as management events. For more information about S3 API operations, see the Amazon S3 API Reference.

How CloudTrail captures requests made to Amazon S3

By default, CloudTrail logs S3 bucket-level API calls that were made in the last 90 days, but not log requests made to objects. Bucket-level calls include events such as `CreateBucket`,

`DeleteBucket`, `PutBucketLifecycle`, `PutBucketPolicy`, and so on. You can see bucket-level events on the CloudTrail console. However, you can't view data events (Amazon S3 object-level calls) there—you must parse or query CloudTrail logs for them.

If you are logging data activity with AWS CloudTrail, the event record for an Amazon S3 `DeleteObjects` data event includes both the `DeleteObjects` event and a `DeleteObject` event for each object deleted as part of that operation. You can exclude the additional visibility about deleted objects from the event record. For more information, see AWS CLI examples for filtering data events in the *AWS CloudTrail User Guide.*

The `InitiateReplication` event includes `objectAnnotationCount` in the `additionalEventData` field. This value indicates the number of annotations on the source object at the time replication was initiated.

When you call `PutObjectRetention`, the CloudTrail data event is recorded with the `eventName` `PutObjectLockRetention`, and includes `objectRetentionInfo` in `additionalEventData`. This field contains the resulting retention mode, retain-until-date, event hold status, and event hold duration.

Amazon S3 account-level actions tracked by CloudTrail logging

CloudTrail logs account-level actions. Amazon S3 records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

The tables in this section list the Amazon S3 account-level actions that are supported for logging by CloudTrail.

Amazon S3 account-level API actions tracked by CloudTrail logging appear as the following event names. The CloudTrail event names differ from the API action name. For example, DeletePublicAccessBlock is DeleteAccountPublicAccessBlock.

- DeleteAccountPublicAccessBlock
- GetAccountPublicAccessBlock
- PutAccountPublicAccessBlock

Amazon S3 bucket-level actions that are tracked by CloudTrail logging

By default, CloudTrail logs bucket-level actions for general purpose buckets. Amazon S3 records are written together with other AWS service records in a log file. CloudTrail determines when to create and write to a new file based on a time period and file size.

This section lists the Amazon S3 bucket-level actions that are supported for logging by CloudTrail.

Amazon S3 bucket-level API actions tracked by CloudTrail logging appear as the following event names. In some cases, the CloudTrail event name differs from the API action name. For example, `PutBucketLifecycleConfiguration` is `PutBucketLifecycle`.

- CreateBucket
- CreateBucketMetadataConfiguration (V2 API operation)
- CreateBucketMetadataTableConfiguration (V1 API operation)
- DeleteBucket
- DeleteBucketAnalyticsConfiguration
- DeleteBucketCors
- DeleteBucketEncryption
- DeleteBucketIntelligentTieringConfiguration
- DeleteBucketInventoryConfiguration
- DeleteBucketLifecycle
- DeleteBucketMetadataConfiguration (V2 API operation)
- DeleteBucketMetadataTableConfiguration (V1 API operation)
- DeleteBucketMetricsConfiguration
- DeleteBucketOwnershipControls
- DeleteBucketPolicy
- DeleteBucketPublicAccessBlock
- DeleteBucketReplication
- DeleteBucketTagging
- GetAccelerateConfiguration
- GetBucketAcl
- GetBucketAnalyticsConfiguration
- GetBucketCors

- GetBucketEncryption
- GetBucketIntelligentTieringConfiguration
- GetBucketInventoryConfiguration
- GetBucketLifecycle
- GetBucketLocation
- GetBucketLogging
- GetBucketMetadataConfiguration (V2 API operation)
- GetBucketMetadataTableConfiguration (V1 API operation)
- GetBucketMetricsConfiguration
- GetBucketNotification
- GetBucketObjectLockConfiguration
- GetBucketOwnershipControls
- GetBucketPolicy
- GetBucketPolicyStatus
- GetBucketPublicAccessBlock
- GetBucketReplication
- GetBucketRequestPayment
- GetBucketTagging
- GetBucketVersioning
- GetBucketWebsite
- HeadBucket
- ListBuckets
- PutAccelerateConfiguration
- PutBucketAcl
- PutBucketAnalyticsConfiguration
- PutBucketCors
- PutBucketEncryption
- PutBucketIntelligentTieringConfiguration
- PutBucketInventoryConfiguration
- PutBucketLifecycle

- PutBucketLogging
- PutBucketMetricsConfiguration
- PutBucketNotification
- PutBucketObjectLockConfiguration
- PutBucketOwnershipControls
- PutBucketPolicy
- PutBucketPublicAccessBlock
- PutBucketReplication
- PutBucketRequestPayment
- PutBucketTagging
- PutBucketVersioning
- PutBucketWebsite
- UpdateBucketMetadataJournalTableConfiguration
- UpdateBucketMetadataInventoryTableConfiguration
- UpdateBucketMetadataAnnotationTableConfiguration

In addition to these API operations, you can also use the OPTIONS object object-level action. This action is treated like a bucket-level action in CloudTrail logging because the action checks the CORS configuration of a bucket.

> **Note**
>
> The HeadBucket API is supported as an Amazon S3 data event in CloudTrail.

Amazon S3 Express One Zone bucket-level (Regional API endpoint) actions tracked by CloudTrail logging

By default, CloudTrail logs bucket-level actions for directory buckets as management events. The `eventsource` for CloudTrail management events for S3 Express One Zone is `s3express.amazonaws.com`.

These following Regional endpoint API operations are logged to CloudTrail.

- CreateBucket

- DeleteBucket
- DeleteBucketPolicy
- GetBucketPolicy
- PutBucketPolicy
- ListDirectoryBuckets
- PutBucketEncryption
- GetBucketEncryption
- DeleteBucketEncryption

For more information, see Logging with AWS CloudTrail for S3 Express One Zone

Amazon S3 object-level actions in cross-account scenarios

The following are special use cases involving the object-level API calls in cross-account scenarios and how CloudTrail logs are reported. CloudTrail delivers logs to the requester (the account that made the API call), except in some access denied cases where log entries are redacted or omitted. When setting up cross-account access, consider the examples in this section.

> **Note**
>
> The examples assume that CloudTrail logs are appropriately configured.

Example 1: CloudTrail delivers logs to the bucket owner

CloudTrail delivers logs to the bucket owner even if the bucket owner does not have permissions for the same object API operation. Consider the following cross-account scenario:

- Account A owns the bucket.
- Account B (the requester) tries to access an object in that bucket.
- Account C owns the object. Account C might or might not be the same account as Account A.

> **Note**
>
> CloudTrail always delivers object-level API logs to the requester (Account B). In addition, CloudTrail also delivers the same logs to the bucket owner (Account A) even when the

bucket owner does not own the object (Account C) or have permissions for those same API operations on that object.

Example 2: CloudTrail does not proliferate email addresses that are used in setting object ACLs

Consider the following cross-account scenario:

- Account A owns the bucket.
- Account B (the requester) sends a request to set an object ACL grant by using an email address. For more information about ACLs, see Access control list (ACL) overview.

The requester gets the logs along with the email information. However, the bucket owner—if they are eligible to receive logs, as in example 1—gets the CloudTrail log reporting the event. However, the bucket owner doesn't get the ACL configuration information, specifically the grantee email address and the grant. The only information that the log tells the bucket owner is that an ACL API call was made by Account B.

CloudTrail log file entries for Amazon S3 and S3 on Outposts

Important

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

An event represents a single request from any source and includes information about the requested API operation, the date and time of the operation, request parameters, and so on. CloudTrail log files aren't an ordered stack trace of the public API calls, so events don't appear in any specific order.

> **Note**
>
> To view CloudTrail log file examples for Amazon S3 Express One Zone, see CloudTrail log file examples for S3 Express One Zone.

For more information, see the following examples.

Topics

- Example: CloudTrail log file entry for Amazon S3

Example: CloudTrail log file entry for Amazon S3

The following example shows a CloudTrail log entry that demonstrates the GET Service, PutBucketAcl, and GetBucketVersioning actions.

```
{
    "Records": [
    {
        "eventVersion": "1.03",
        "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
        "eventTime": "2019-02-01T03:18:19Z",
        "eventSource": "s3.amazonaws.com",
        "eventName": "ListBuckets",
        "awsRegion": "us-west-2",
        "sourceIPAddress": "127.0.0.1",
        "userAgent": "[]",
        "requestParameters": {
            "host": [
                "s3.us-west-2.amazonaws.com"
            ]
        },
        "responseElements": null,
        "additionalEventData": {
```

```
            "SignatureVersion": "SigV2",
            "AuthenticationMethod": "QueryString",
            "aclRequired": "Yes"
    },
        "requestID": "47B8E8D397DCE7A6",
        "eventID": "cdc4b7ed-e171-4cef-975a-ad829d4123e8",
        "eventType": "AwsApiCall",
        "recipientAccountId": "444455556666",
        "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    },
    {
       "eventVersion": "1.03",
       "userIdentity": {
            "type": "IAMUser",
            "principalId": "111122223333",
            "arn": "arn:aws:iam::111122223333:user/myUserName",
            "accountId": "111122223333",
            "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
            "userName": "myUserName"
        },
      "eventTime": "2019-02-01T03:22:33Z",
      "eventSource": "s3.amazonaws.com",
      "eventName": "PutBucketAcl",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "",
      "userAgent": "[]",
      "requestParameters": {
          "bucketName": "",
          "AccessControlPolicy": {
              "AccessControlList": {
                  "Grant": {
                      "Grantee": {
                          "xsi:type": "CanonicalUser",
                          "xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
                          "ID":
"d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
                       },
                      "Permission": "FULL_CONTROL"
                  }
              },
```

```
            "xmlns": "http://s3.amazonaws.com/doc/2006-03-01/",
            "Owner": {
                "ID":
"d25639fbe9c19cd30a4c0f43fbf00e2d3f96400a9aa8dabfbbebe1906Example"
            }
        },
        "host": [
            "s3.us-west-2.amazonaws.com"
        ],
        "acl": [
            ""
        ]
    },
    "responseElements": null,
    "additionalEventData": {
        "SignatureVersion": "SigV4",
        "CipherSuite": "ECDHE-RSA-AES128-SHA",
        "AuthenticationMethod": "AuthHeader"
    },
    "requestID": "BD8798EACDD16751",
    "eventID": "607b9532-1423-41c7-b048-ec2641693c47",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333",
    "tlsDetails": {
          "tlsVersion": "TLSv1.2",
          "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
          "clientProvidedHostHeader": "s3.amazonaws.com"
  }
  },
  {
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "111122223333",
        "arn": "arn:aws:iam::111122223333:user/myUserName",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "myUserName"
      },
    "eventTime": "2019-02-01T03:26:37Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "GetBucketVersioning",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "",
```

```
      "userAgent": "[]",
      "requestParameters": {
          "host": [
              "s3.us-west-2.amazonaws.com"
          ],
          "bucketName": "amzn-s3-demo-bucket1",
          "versioning": [
              ""
          ]
      },
      "responseElements": null,
      "additionalEventData": {
          "SignatureVersion": "SigV4",
          "CipherSuite": "ECDHE-RSA-AES128-SHA",
          "AuthenticationMethod": "AuthHeader"
    },
      "requestID": "07D681279BD94AED",
      "eventID": "f2b287f3-0df1-4961-a2f4-c4bdfed47657",
      "eventType": "AwsApiCall",
      "recipientAccountId": "111122223333",
      "tlsDetails": {
            "tlsVersion": "TLSv1.2",
            "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
            "clientProvidedHostHeader": "s3.amazonaws.com"
    }
    }
  ]
}
```


Enabling CloudTrail event logging for S3 buckets and objects

You can use CloudTrail data events to get information about bucket and object-level requests in Amazon S3. To enable CloudTrail data events for all of your buckets or for a list of specific buckets, you must create a trail manually in CloudTrail.

Note

- The default setting for CloudTrail is to find only management events. Check to ensure that you have the data events enabled for your account.

- With an S3 bucket that is generating a high workload, you could quickly generate thousands of logs in a short amount of time. Be mindful of how long you choose to enable CloudTrail data events for a busy bucket.

CloudTrail stores Amazon S3 data event logs in an S3 bucket of your choosing. Consider using a bucket in a separate AWS account to better organize events from multiple buckets that you might own into a central place for easier querying and analysis. AWS Organizations helps you create an AWS account that is linked to the account that owns the bucket that you're monitoring. For more information, see What is AWS Organizations? in the *AWS Organizations User Guide*.

When you log data events for a trail in CloudTrail, you can choose to use advanced event selectors or basic event selectors to log data events for objects stored in general purpose buckets. To log data events for objects stored in directory buckets, you must use advanced event selectors. For more information, see Logging with AWS CloudTrail for S3 Express One Zone.

When you create a trail in the CloudTrail console using advanced event selectors, in the data events section, you can choose **Log all events** for the **Log selector template** to log all object-level events. When you create a trail in the CloudTrail console using basic event selectors, in the data events section, you can select the **Select all S3 buckets in your account** check box to log all object-level events.

Note

- It's a best practice to create a lifecycle configuration for your AWS CloudTrail data event bucket. Configure the lifecycle configuration to periodically remove log files after the period of time you believe you need to audit them. Doing so reduces the amount of data that Athena analyzes for each query. For more information, see Setting an S3 Lifecycle configuration on a bucket.
- For more information about logging format, see Logging Amazon S3 API calls using AWS CloudTrail.
- For examples of how to query CloudTrail logs, see the *AWS Big Data Blog* post Analyze Security, Compliance, and Operational Activity Using AWS CloudTrail and Amazon Athena.

Enable logging for objects in a bucket using the console

You can use the AWS CloudTrail console to configure a CloudTrail trail to log data events for objects in an S3 bucket. CloudTrail supports logging Amazon S3 object-level API operations such as `GetObject`, `DeleteObject`, and `PutObject`. These events are called *data events*.

By default, CloudTrail trails don't log data events, but you can configure trails to log data events for S3 buckets that you specify, or to log data events for all the Amazon S3 buckets in your AWS account. For more information, see Logging Amazon S3 API calls using AWS CloudTrail.

CloudTrail does not populate data events in the CloudTrail event history. Additionally, not all bucket-level actions are populated in the CloudTrail event history. For more information about the Amazon S3 bucket–level API actions tracked by CloudTrail logging, see Amazon S3 bucket-level actions that are tracked by CloudTrail logging. For more information about how to query CloudTrail logs, see the AWS Knowledge Center article about using Amazon CloudWatch Logs filter patterns and Amazon Athena to query CloudTrail logs.

> **Note**
>
> If you are logging data activity with AWS CloudTrail, the event record for an Amazon S3 `DeleteObjects` data event includes both the `DeleteObjects` event and a `DeleteObject` event for each object deleted as part of that operation. You can exclude the additional visibility about deleted objects from the event record. For more information, see AWS CLI examples for filtering data events in the *AWS CloudTrail User Guide*.

To enable CloudTrail data events logging for objects in an S3 general purpose bucket or an S3 directory bucket see Creating a trail with the CloudTrail console in the *AWS CloudTrail User Guide*.

For more information about logging objects in an S3 directory bucket, see Logging with AWS CloudTrail for directory buckets.

For information about using the CloudTrail console to configure a trail to log S3 data events, see Logging data events in the *AWS CloudTrail User Guide*.

To disable CloudTrail data events logging for objects in an S3 bucket, see Deleting a trail with the CloudTrail console in the *AWS CloudTrail User Guide*.

Important

Additional charges apply for data events. For more information, see AWS CloudTrail pricing.

For more information about CloudTrail logging with S3 buckets, see the following topics:

- Creating a general purpose bucket
- Viewing the properties for an S3 general purpose bucket
- Logging Amazon S3 API calls using AWS CloudTrail
- Working with CloudTrail log files in the *AWS CloudTrail User Guide*

Identifying Amazon S3 requests using CloudTrail

In Amazon S3, you can identify requests using an AWS CloudTrail event log. AWS CloudTrail is the preferred way of identifying Amazon S3 requests, but if you are using Amazon S3 server access logs, see the section called "Identifying S3 requests".

Topics

- Identifying requests made to Amazon S3 in a CloudTrail log
- Identifying Amazon S3 Signature Version 2 requests by using CloudTrail
- Identifying access to S3 objects by using CloudTrail

Identifying requests made to Amazon S3 in a CloudTrail log

After you set up CloudTrail to deliver events to a bucket, you should start to see objects go to your destination bucket on the Amazon S3 console. These are formatted as follows:

`s3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/Region/yyyy/mm/dd`

Events logged by CloudTrail are stored as compressed, gzipped JSON objects in your S3 bucket. To efficiently find requests, you should use a service like Amazon Athena to index and query the CloudTrail logs.

For more information about CloudTrail and Athena, see Creating the table for AWS CloudTrail logs in Athena using partition projection in the *Amazon Athena User Guide*.

Identifying Amazon S3 Signature Version 2 requests by using CloudTrail

You can use a CloudTrail event log to identify which API signature version was used to sign a request in Amazon S3. This capability is important because support for Signature Version 2 will be turned off (deprecated). After that, Amazon S3 will no longer accept requests that use Signature Version 2, and all requests must use *Signature Version 4* signing.

We *strongly* recommend that you use CloudTrail to help determine whether any of your workflows are using Signature Version 2 signing. Remediate them by upgrading your libraries and code to use Signature Version 4 instead to prevent any impact to your business.

For more information, see Announcement: AWS CloudTrail for Amazon S3 adds new fields for enhanced security auditing in AWS re:Post.

> **Note**
>
> CloudTrail events for Amazon S3 include the signature version in the request details under the key name of `'additionalEventData`. To find the signature version on requests made for objects in Amazon S3 such as `GET`, `PUT`, and `DELETE` requests, you must enable CloudTrail data events. (This feature is turned off by default.)

AWS CloudTrail is the preferred method for identifying Signature Version 2 requests. If you're using Amazon S3 server-access logs, see Identifying Signature Version 2 requests.

Topics

- Athena query examples for identifying Amazon S3 Signature Version 2 requests
- Partitioning Signature Version 2 data

Athena query examples for identifying Amazon S3 Signature Version 2 requests

Example— Select all Signature Version 2 events, and print only EventTime, S3_Action, Request_Parameters, Region, SourceIP, and UserAgent

In the following Athena query, replace *`s3_cloudtrail_events_db.cloudtrail_table`* with your Athena details, and increase or remove the limit as needed.

```
SELECT EventTime, EventName as S3_Action, requestParameters as Request_Parameters,
 awsregion as AWS_Region, sourceipaddress as Source_IP, useragent as User_Agent
FROM s3_cloudtrail_events_db.cloudtrail_table
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
LIMIT 10;
```

Example— Select all requesters that are sending Signature Version 2 traffic

```
SELECT useridentity.arn, Count(requestid) as RequestCount
FROM s3_cloudtrail_events_db.cloudtrail_table
WHERE eventsource='s3.amazonaws.com'
    and json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
Group by useridentity.arn
```

Partitioning Signature Version 2 data

If you have a large amount of data to query, you can reduce the costs and running time of Athena by creating a partitioned table.

To do this, create a new table with partitions as follows.

```
CREATE EXTERNAL TABLE s3_cloudtrail_events_db.cloudtrail_table_partitioned(
     eventversion STRING,
     userIdentity STRUCT<
         type:STRING,
         principalid:STRING,
         arn:STRING,
         accountid:STRING,
         invokedby:STRING,
         accesskeyid:STRING,
         userName:STRING,
       sessioncontext:STRUCT<
                  attributes:STRUCT<
                  mfaauthenticated:STRING,
                  creationdate:STRING>,
                  sessionIssuer:STRUCT<
                  type:STRING,
```

```
                    principalId:STRING,
                    arn:STRING,
                    accountId:STRING,
                    userName:STRING>
                >
            >,
        eventTime STRING,
        eventSource STRING,
        eventName STRING,
        awsRegion STRING,
        sourceIpAddress STRING,
        userAgent STRING,
        errorCode STRING,
        errorMessage STRING,
        requestParameters STRING,
        responseElements STRING,
        additionalEventData STRING,
        requestId STRING,
        eventId STRING,
        resources ARRAY<STRUCT<ARN:STRING,accountId: STRING,type:STRING>>,
        eventType STRING,
        apiVersion STRING,
        readOnly STRING,
        recipientAccountId STRING,
        serviceEventDetails STRING,
        sharedEventID STRING,
        vpcEndpointId STRING
    )
    PARTITIONED BY (region string, year string, month string, day string)
    ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
    STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
    OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
    LOCATION 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/';
```

Then, create the partitions individually. You can't get results from dates that you haven't created.

```
ALTER TABLE s3_cloudtrail_events_db.cloudtrail_table_partitioned ADD
    PARTITION (region= 'us-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-east-1/2019/02/19/'
    PARTITION (region= 'us-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-west-1/2019/02/19/'
```

```
    PARTITION (region= 'us-west-2', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/us-west-2/2019/02/19/'
    PARTITION (region= 'ap-southeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-southeast-1/2019/02/19/'
    PARTITION (region= 'ap-southeast-2', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-southeast-2/2019/02/19/'
    PARTITION (region= 'ap-northeast-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/ap-northeast-1/2019/02/19/'
    PARTITION (region= 'eu-west-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/eu-west-1/2019/02/19/'
    PARTITION (region= 'sa-east-1', year= '2019', month= '02', day= '19') LOCATION
 's3://amzn-s3-demo-bucket1/AWSLogs/111122223333/CloudTrail/sa-east-1/2019/02/19/';
```

You can then make the request based on these partitions, and you don't need to load the full bucket.

```
SELECT useridentity.arn,
Count(requestid) AS RequestCount
FROM s3_cloudtrail_events_db.cloudtrail_table_partitioned
WHERE eventsource='s3.amazonaws.com'
AND json_extract_scalar(additionalEventData, '$.SignatureVersion')='SigV2'
AND region='us-east-1'
AND year='2019'
AND month='02'
AND day='19'
Group by useridentity.arn
```

Identifying access to S3 objects by using CloudTrail

You can use your AWS CloudTrail event logs to identify Amazon S3 object access requests for data events such as `GetObject`, `DeleteObject`, and `PutObject`, and discover additional information about those requests.

> **Note**
>
> If you are logging data activity with AWS CloudTrail, the event record for an Amazon S3 `DeleteObjects` data event includes both the `DeleteObjects` event and a `DeleteObject` event for each object deleted as part of that operation. You can exclude

the additional visibility about deleted objects from the event record. For more information, see AWS CLI examples for filtering data events in the *AWS CloudTrail User Guide.*

The following example shows how to get all PUT object requests for Amazon S3 from an AWS CloudTrail event log.

Topics

- Athena query examples for identifying Amazon S3 object access requests

Athena query examples for identifying Amazon S3 object access requests

In the following Athena query examples, replace *s3_cloudtrail_events_db.cloudtrail_table* with your Athena details, and modify the date range as needed.

Example— Select all events that have PUT object access requests, and print only EventTime, EventSource, SourceIP, UserAgent, BucketName, object, and UserARN

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  eventName = 'PutObject'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example— Select all events that have GET object access requests, and print only EventTime, EventSource, SourceIP, UserAgent, BucketName, object, and UserARN

```
SELECT
  eventTime,
```

```
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  userIdentity.arn as userArn
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  eventName = 'GetObject'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example— Select all anonymous requester events to a bucket in a certain period and print only EventTime, EventName, EventSource, SourceIP, UserAgent, BucketName, UserARN, and AccountID

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  userIdentity.arn as userArn,
  userIdentity.accountId
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  userIdentity.accountId = 'anonymous'
  AND eventTime BETWEEN '2019-07-05T00:00:00Z' and '2019-07-06T00:00:00Z'
```

Example— Identify all requests that required an ACL for authorization

The following Amazon Athena query example shows how to identify all requests to your S3 buckets that required an access control list (ACL) for authorization. If the request required an ACL for authorization, the `aclRequired` value in `additionalEventData` is `Yes`. If no ACLs were required, `aclRequired` is not present. You can use this information to migrate those ACL permissions to the appropriate bucket policies. After you've created these bucket policies, you can disable ACLs for these buckets. For more information about disabling ACLs, see Prerequisites for disabling ACLs.

```
SELECT
  eventTime,
  eventName,
  eventSource,
  sourceIpAddress,
  userAgent,
  userIdentity.arn as userArn,
  json_extract_scalar(requestParameters, '$.bucketName') as bucketName,
  json_extract_scalar(requestParameters, '$.key') as object,
  json_extract_scalar(additionalEventData, '$.aclRequired') as aclRequired
FROM
  s3_cloudtrail_events_db.cloudtrail_table
WHERE
  json_extract_scalar(additionalEventData, '$.aclRequired') = 'Yes'
  AND eventTime BETWEEN '2022-05-10T00:00:00Z' and '2022-08-10T00:00:00Z'
```

Note

- These query examples can also be useful for security monitoring. You can review the results for `PutObject` or `GetObject` calls from unexpected or unauthorized IP addresses or requesters and for identifying any anonymous requests to your buckets.
- This query only retrieves information from the time at which logging was enabled.

If you are using Amazon S3 server access logs, see Identifying object access requests.

Logging requests with server access logging

Server access logging provides detailed records for the requests that are made to a bucket. Server access logs can be used for operational analysis, to understand your security posture, and to charge back users.

Amazon S3 periodically collects access log records, consolidates them, and delivers them to your chosen destination. You can deliver logs to Amazon CloudWatch Logs or to an Amazon S3 general purpose bucket.

> **Note**
>
> Server access logs don't record information about wrong-Region redirect errors for Regions that launched after March 20, 2019. Wrong-Region redirect errors occur when a request for an object or bucket is made outside the Region in which the bucket exists.

Delivery options

You can deliver server access logs to two destinations. You can use one or both.

- **Amazon CloudWatch Logs** – Delivers logs in structured format to a CloudWatch Logs log group. You can query logs with CloudWatch Logs Insights, aggregate logs across accounts and Regions, and encrypt logs with AWS KMS. From the log group, you can also deliver logs to Amazon S3 in JSON or Apache Parquet format, or mirror logs to S3 Tables in Apache Iceberg format for SQL analytics. CloudWatch vended logs ingestion rates apply.
- **Amazon S3 general purpose bucket** – Delivers logs in space-delimited text format to an Amazon S3 bucket you specify. There is no charge for log delivery. You pay only for the storage of the log files. The destination bucket must be in the same AWS Region and AWS account as the source bucket.

The following table compares the two delivery options.

Comparing server access log delivery options

Capability	Amazon CloudWatch Logs	Amazon S3 general purpose bucket
Destinations	CloudWatch Logs log group, Amazon S3 (JSON or Parquet), or Amazon Data Firehose. The CloudWatch Logs log group delivery can be extended to mirror logs to S3 Tables in Iceberg format.	Amazon S3 bucket (space-de limited text)
Log format	Structured JSON (log group), JSON or Apache Parquet	Space-delimited text

Capability	Amazon CloudWatch Logs	Amazon S3 general purpose bucket
	(Amazon S3), Apache Iceberg (S3 Tables)	
Querying	CloudWatch Logs Insights (log group), Amazon Athena or Spark (S3 Tables)	Amazon Athena
SQL analytics through S3 Tables	Yes (automatic Iceberg mirror)	No (requires ETL)
Cross-account aggregation	Yes (Amazon S3 and Firehose destinations support cross-account delivery)	No (same account only)
Cross-Region aggregation	Yes (Amazon S3 and Firehose destinations support cross-Region delivery)	No (same Region only)
AWS KMS encryption	Yes	No (SSE-S3 only)
Setup	CloudWatch Logs APIs or Amazon S3 console	Amazon S3 API (`PutBucketLogging`) or console
Delivery latency	Within a few hours	Within a few hours

Best-effort server log delivery

Server access log records are delivered on a best-effort basis. Most requests for a bucket that is properly configured for logging result in a delivered log record. Most log records are delivered within a few hours of the time that they are recorded, but they can be delivered more frequently.

The completeness and timeliness of server logging is not guaranteed. The log record for a particular request might be delivered long after the request was actually processed, or *it might not be delivered at all*. It is possible that you might even see a duplication of a log record. The purpose of server logs is to give you an idea of the nature of traffic against your bucket. Although log

records are rarely lost or duplicated, be aware that server logging is not meant to be a complete accounting of all requests.

Because of the best-effort nature of server logging, your usage reports might include one or more access requests that do not appear in a delivered server log. You can find these usage reports under **Cost & usage reports** in the AWS Billing and Cost Management console.

Bucket logging status changes take effect over time

Changes to the logging status of a bucket take time to actually affect the delivery of log files. For example, if you enable logging for a bucket, some requests made in the following hour might be logged, and others might not. Suppose that you change the destination bucket for logging from bucket A to bucket B. For the next hour, some logs might continue to be delivered to bucket A, whereas others might be delivered to the new destination bucket B. Similarly, if you change the CloudWatch Logs log group used for delivery, some logs might continue to be delivered to the previous log group during the transition. In all cases, the new settings eventually take effect without any further action on your part.

Pricing

The cost of server access logging depends on which delivery option you use.

- **Amazon CloudWatch Logs** – You pay for ingestion at CloudWatch vended logs rates, with volume-based tiered pricing. CloudWatch Logs compresses logs before storage, so stored volume is significantly less than ingested volume. If you enable the S3 Tables mirror, there is no additional charge for storage or table maintenance. You pay only for query requests at S3 Tables pricing. For current rates, see CloudWatch pricing.
- **Amazon S3 general purpose bucket** – There is no charge for log delivery. You pay only for the storage of the log files at standard Amazon S3 storage rates. We do not assess data-transfer charges for log file delivery, but we do charge the normal data-transfer rate for accessing the log files. For current rates, see Amazon S3 pricing.

For more information about delivering and using server access logs, see the following sections.

Delivering logs to Amazon CloudWatch Logs

When you deliver server access logs to Amazon CloudWatch Logs, you get structured JSON logs that you can query interactively with CloudWatch Logs Insights, aggregate across accounts and

Regions, and optionally mirror to S3 Tables in Apache Iceberg format for SQL analytics. You can also deliver logs to Amazon S3 in JSON or Apache Parquet format or route them through Amazon Data Firehose.

Topics

- Delivering server access logs to CloudWatch Logs
- Log format in CloudWatch Logs
- Managing log retention
- Querying logs with CloudWatch Logs Insights
- Querying access logs in S3 Tables
- Troubleshooting CloudWatch Logs delivery

Delivering server access logs to CloudWatch Logs

You can deliver Amazon S3 server access logs to a CloudWatch Logs log group using Amazon CloudWatch Logs vended log delivery. Once logs are in CloudWatch Logs, you can query them with CloudWatch Logs Insights, aggregate them across accounts and Regions, encrypt them with AWS Key Management Service (AWS KMS), and optionally deliver them to Amazon S3 Tables in Apache Iceberg format for SQL analytics. You can also deliver logs to Amazon S3 in JSON or Apache Parquet format or route them through Amazon Data Firehose.

You configure delivery using CloudWatch Logs APIs, not Amazon S3 APIs. You specify a log group for each source bucket. Multiple buckets can share the same log group, or each bucket can deliver to its own.

Tip

You can set up multiple destinations for the same bucket. For example, deliver to a CloudWatch Logs log group for interactive querying and simultaneously deliver to Amazon S3 in Parquet format for long-term retention at lower cost.

Note

CloudWatch Logs vended log delivery also supports Amazon Data Firehose and Amazon S3 as delivery destinations, including Apache Parquet output format. This page covers the CloudWatch Logs log group destination, which also supports the S3 Tables integration for

SQL analytics. For information about Firehose and Amazon S3 destinations, see Enable logging from AWS services in the *Amazon CloudWatch Logs User Guide*.

Note

For information about delivering server access logs to an Amazon S3 general purpose bucket instead, see Enabling Amazon S3 server access logging.

Prerequisites

Before you set up delivery, you need the following:

- An Amazon S3 general purpose bucket (the source bucket you want to log).
- A CloudWatch Logs log group in the same AWS Region as the source bucket.
- IAM permissions for the caller. At a minimum, you need the following permissions:
  - `logs:PutDeliverySource`
  - `logs:PutDeliveryDestination`
  - `logs:CreateDelivery`
  - `s3:AllowVendedLogDeliveryForResource` (on the source bucket)

  If you plan to enable the S3 Tables integration, you also need:
  - `observabilityadmin:CreateS3TableIntegration`
  - `logs:AssociateSourceToS3TableIntegration`
  - `s3tables:CreateTableBucket`
  - `s3tables:PutTableBucketEncryption`
  - `s3tables:PutTableBucketPolicy`

Setting up delivery (AWS CLI)

To set up server access log delivery to CloudWatch Logs using the AWS Command Line Interface, complete the following three steps.

1. **Create a delivery source.** This registers your Amazon S3 bucket as a log source with CloudWatch Logs.

```
aws logs put-delivery-source \
  --name my-sal-source \
  --resource-arn arn:aws:s3:::my-bucket \
  --log-type S3_SERVER_ACCESS_LOGS
```

2. **Create a delivery destination.** This specifies the CloudWatch Logs log group where logs will be delivered.

```
aws logs put-delivery-destination \
  --name my-sal-destination \
  --delivery-destination-configuration '{"destinationResourceArn":
 "arn:aws:logs:us-east-1:123456789012:log-group:my-sal-logs"}'
```

3. **Create a delivery.** This links the source to the destination and starts log delivery.

```
aws logs create-delivery \
  --delivery-source-name my-sal-source \
  --delivery-destination-arn arn:aws:logs:us-east-1:123456789012:delivery-
destination:my-sal-destination
```

After you complete these steps, Amazon S3 begins delivering server access logs to your CloudWatch Logs log group. The configuration takes 15–20 minutes to propagate. After propagation, log delivery typically begins within a few hours. Only new requests are logged – there is no backfill of historical data.

Setting up delivery (Amazon S3 console)

1. Sign in to the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. Choose the name of the source bucket that you want to enable logging for.
4. Choose the **Properties** tab.
5. In the **Server access log delivery to CloudWatch Logs** section, choose **Edit**.
6. Choose **Enable**.
7. Select or create a CloudWatch Logs log group as the delivery destination.
8. (Optional) To deliver logs to S3 Tables in Apache Iceberg format, select **Enable S3 Tables integration**.
9. Choose **Save changes**.

After you save, the **Server access log delivery to CloudWatch Logs** section displays the enabled log group.

Setting up delivery (CloudFormation)

You can use AWS CloudFormation to automate the setup of server access log delivery to CloudWatch Logs. The following template creates the required resources: an IAM role, a CloudWatch Logs log group, a delivery source, a delivery destination, and a delivery.

```
AWSTemplateFormatVersion: '2010-09-09'
Description: S3 Access Logs delivery to CloudWatch Logs

Parameters:
  SourceBucketArn:
    Type: String
    Description: ARN of the S3 bucket to enable access logging
  LogGroupName:
    Type: String
    Default: /aws/s3/access-logs
    Description: CloudWatch Log Group name for SAL delivery

Resources:
  SALLogGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: !Ref LogGroupName
      RetentionInDays: 180

  SALDeliverySource:
    Type: AWS::Logs::DeliverySource
    Properties:
      Name: !Sub '${AWS::StackName}-sal-source'
      ResourceArn: !Ref SourceBucketArn
      LogType: S3_SERVER_ACCESS_LOGS

  SALDeliveryDestination:
    Type: AWS::Logs::DeliveryDestination
    Properties:
      Name: !Sub '${AWS::StackName}-sal-destination'
      DestinationResourceArn: !GetAtt SALLogGroup.Arn

  SALDelivery:
    Type: AWS::Logs::Delivery
```

```
    Properties:
      DeliverySourceName: !Ref SALDeliverySource
      DeliveryDestinationArn: !GetAtt SALDeliveryDestination.Arn
```

Enabling the S3 Tables integration (optional)

When you enable the S3 Tables integration, CloudWatch Logs automatically delivers your server access logs in Apache Iceberg format to the `aws-cloudwatch` managed table bucket in your account. You can then query this data using Amazon Athena, Amazon Redshift, or any Iceberg-compatible tool.

There is no additional charge for storage or table maintenance in the S3 Tables integration. You pay only for CloudWatch Logs ingestion and for query requests at S3 Tables pricing.

1. **Create the S3 Tables integration** (once per account per Region).

   ```
   aws observabilityadmin create-s3-table-integration \
     --role-arn arn:aws:iam::123456789012:role/CWLogsS3TableIntegrationRole \
     --encryption '{"SseAlgorithm": "AES256"}'
   ```

   The IAM role must trust the `logs.amazonaws.com` service principal and grant the `logs:integrateWithS3Table` permission. The following is an example trust policy:

   ```
         {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "logs.amazonaws.com"
         },
         "Action": "sts:AssumeRole",
         "Condition": {
           "StringEquals": {
             "aws:SourceAccount": "123456789012"
           },
           "ArnLike": {
             "aws:SourceArn": "arn:aws:logs:us-east-1:123456789012:log-group:log-
   group-name"
           }
         }
       }
   ```

```
  ]
}
```

The role also needs the following permission policy:

```
      {
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "logs:integrateWithS3Table",
      "Resource": "arn:aws:logs:us-east-1:123456789012:log-group:log-group-name",
      "Condition": {
        "StringEquals": {
          "aws:ResourceAccount": "123456789012"
        }
      }
    }
  ]
}
```

2. **Associate your delivery source to the integration.**

```
aws logs associate-source-to-s3-table-integration \
  --integration-arn integration-arn \
  --data-source '{"name":"amazon_s3","type":"server_access"}'
```

After these steps, CloudWatch Logs creates the `aws-cloudwatch` managed table bucket and begins delivering logs in Iceberg format. Data typically appears within an hour of the first delivery to CloudWatch Logs.

For information about querying the S3 Tables integration, see Querying access logs in S3 Tables.

Cross-account and cross-Region aggregation

The CloudWatch Logs delivery path supports cross-account and cross-Region log aggregation with two options:

- **CloudWatch cross-account observability** allows you to query logs in place across accounts without copying data.

- **CloudWatch Logs centralization** allows you to consolidate logs from multiple accounts and Regions into a central log group, optionally using subscription filters for selective forwarding.

For multi-account setups, administrators can use CloudWatch Telemetry Config enablement rules with AWS Organizations to automatically enable server access log delivery across member accounts.

> **Note**
>
> Enablement rules do not currently support automatic enablement of the S3 Tables integration. You must enable the integration separately in each account.

Encryption

You can encrypt server access logs delivered to CloudWatch Logs by configuring AWS KMS encryption on your log group. When server access logs are delivered to that log group, they are encrypted with your AWS KMS key. This addresses a key limitation of the Amazon S3 general purpose bucket delivery path, which supports only SSE-S3 encryption.

If you enable the S3 Tables integration with a AWS KMS-encrypted log group, you must grant additional permissions in your AWS KMS key policy to the following service principals:

```
     {
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "EnableCWSystemTablesKeyUsage",
    "Effect": "Allow",
    "Principal": {
      "Service": "systemtables.cloudwatch.amazonaws.com"
    },
    "Action": [
      "kms:DescribeKey",
      "kms:GenerateDataKey",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-id",
    "Condition": {
      "StringEquals": {
        "aws:SourceAccount": "123456789012"
```

```
        }
      }
    },
    {
      "Sid": "EnableS3TablesMaintenanceKeyUsage",
      "Effect": "Allow",
      "Principal": {
        "Service": "maintenance.s3tables.amazonaws.com"
      },
      "Action": [
        "kms:GenerateDataKey",
        "kms:Decrypt"
      ],
      "Resource": "arn:aws:kms:us-east-1:123456789012:key/key-id",
      "Condition": {
        "StringLike": {
          "kms:EncryptionContext:aws:s3:arn": "table-bucket-arn/*"
        }
      }
    }
  ]
}
```

The `systemtables.cloudwatch.amazonaws.com` principal is used by CloudWatch Logs to write logs to S3 Tables. The `maintenance.s3tables.amazonaws.com` principal is used by S3 Tables for Iceberg operations such as compaction.

Disabling delivery

To stop delivering server access logs to CloudWatch Logs, you can delete the delivery or the delivery source:

- **Delete the delivery** – stops log delivery but keeps the delivery source configuration intact. You can create a new delivery later without reconfiguring the source.

```
aws logs delete-delivery \
  --id delivery-id
```

- **Delete the delivery source** – removes the source configuration entirely. Use this if you no longer want the bucket associated with CloudWatch Logs delivery.

```
aws logs delete-delivery-source \
```

```
--name delivery-source-name
```

Previously delivered logs remain in the log group and in the S3 Tables integration (if enabled). Deleting the delivery does not remove existing log data.

Log format in CloudWatch Logs

When Amazon S3 server access logs are delivered to Amazon CloudWatch Logs, each log record is a structured JSON object. This is different from the space-delimited text format used for delivery to an Amazon S3 general purpose bucket. The structured format makes fields directly queryable in CloudWatch Logs Insights without regex parsing.

> **Note**
>
> For the space-delimited text format used for general purpose bucket delivery, see Amazon S3 server access log format.

Log record fields

The following table describes the fields in each server access log record delivered to CloudWatch Logs.

Server access log fields in CloudWatch Logs

Field	Type	Description
`schema_version_id`	string	Version identifier for the log record schema (for example, `V_1_0`). Use this field to detect schema changes when new fields are added in future versions.
`bucket_arn`	string	ARN of the source bucket.
`bucket_name`	string	Name of the bucket that the request was made to. Derived from the bucket ARN.
`request_time`	string	Time the request was received, in UTC. Formatted as ISO 8601 (`yyyy-MM-dd'T'HH:mm:ss.SSS'Z`

Field	Type	Description
		'). In the S3 Tables integration (Iceberg), this field is stored as `timestamptz` .
`bucket_owner_id`	string	Canonical user ID of the owner of the source bucket.
`remote_ip`	string	Apparent IP address of the requester. Defaults to `null` if not available.
`requester`	string	Canonical user ID of the requester, or `null` for unauthenticated requests. For IAM users and roles, this is the ARN.
`request_id`	string	Unique identifier generated by Amazon S3 for each request. Defaults to `null` if not available.
`operation`	string	The operation that was performed, for example `REST.GET.OBJECT` or `REST.PUT.OBJECT` . Defaults to `null` if not available.
`key_name`	string	Object key in the request, or `null` if the operation does not involve an object.
`request_uri`	string	The `Request-URI` part of the HTTP request message, or `null` if not available.
`http_status`	int	Numeric HTTP status code of the response, for example `200` or `403`. Defaults to `null` if not available.
`error_code`	string	Amazon S3 error code, for example `NoSuchKey` or `AccessDenied` , or `null` if no error occurred.
`bytes_sent_size`	long	Number of response bytes sent, excluding HTTP protocol overhead. Defaults to `null` if not available .

Field	Type	Description
object_size	long	Total size of the object. Defaults to null if not available.
total_duration	long	Total time of the request from the time the request was received to the last byte of the response sent, in milliseconds. Defaults to null if not available.
turn_around_duration	long	Time Amazon S3 spent processing the request, in milliseconds. This value is measured from the time the last byte of the request was received until the time the first byte of the response was sent. Defaults to null if not available.
referer	string	Value of the HTTP Referer header, or null if not present.
user_agent	string	Value of the HTTP User-Agent header. Defaults to null if not available.
version_id	string	Version ID in the request, or null if the operation does not involve a versioned object.
host_id	string	Amazon S3 extended request ID (x-amz-id-2). Defaults to null if not available.
signature_version	string	Signature version used to authenticate the request: SigV2 or SigV4. Defaults to null for unauthenticated requests.
cipher_suite	string	TLS cipher suite negotiated for HTTPS requests, or null for HTTP.
authentication_type	string	Type of request authentication: AuthHeader , QueryString , or null.
host_header	string	Endpoint used to connect to Amazon S3. Defaults to null if not available.

Field	Type	Description
`tls_version`	string	TLS version negotiated by the client: `TLSv1.1`, `TLSv1.2`, `TLSv1.3`, or `null`.
`access_point_arn`	string	ARN of the access point used for the request, or `null` if the request did not use an access point.
`acl_required`	boolean	Whether an ACL was required for the request. Defaults to `null` when not applicable.
`source_region`	string	AWS Region from which the request originated, or `null` when the origin Region cannot be determined. Examples include PrivateLink connections, Direct Connect connections, Bring your own IP addresses (BYOIP), non-AWS IP addresses, and logs generated by policy-based actions such as lifecycle and checksum.

Sample log record

The following is an example of a server access log record as it appears in CloudWatch Logs.

```
{
  "schema_version_id": "V_1_0",
  "bucket_arn": "arn:aws:s3:::my-data-bucket",
  "bucket_name": "my-data-bucket",
  "request_time": "2026-04-29T14:32:16.000Z",
  "bucket_owner_id":
 "79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be",
  "remote_ip": "203.0.113.45",
  "requester": "arn:aws:iam::123456789012:role/DataAnalystRole",
  "request_id": "3E57427F33A59F07",
  "operation": "REST.GET.OBJECT",
  "key_name": "reports/2026/quarterly-summary.parquet",
  "request_uri": "GET /my-data-bucket/reports/2026/quarterly-summary.parquet HTTP/1.1",
  "http_status": 200,
  "error_code": null,
  "bytes_sent_size": 1048576,
  "object_size": 1048576,
```

```
  "total_duration": 45,
  "turn_around_duration": 12,
  "referer": null,
  "user_agent": "aws-sdk-java/2.20.0",
  "version_id": null,
  "host_id": "s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/
VLi31234=",
  "signature_version": "SigV4",
  "cipher_suite": "TLS_AES_128_GCM_SHA256",
  "authentication_type": "AuthHeader",
  "host_header": "my-data-bucket.s3.us-east-1.amazonaws.com",
  "tls_version": "TLSv1.3",
  "access_point_arn": null,
  "acl_required": false,
  "source_region": "us-east-1"
}
```


Differences from general purpose bucket log format

The CloudWatch Logs log format differs from the general purpose bucket log format in the following ways:

- **Format** – CloudWatch Logs logs are structured JSON objects. General purpose bucket logs are space-delimited text.
- **Field names** – The CloudWatch Logs format uses descriptive underscore-separated field names (for example, `bytes_sent_size`, `total_duration`, `key_name`, `authentication_type`). The general purpose bucket format is positional space-delimited text with no named columns.
- **Additional fields** – The CloudWatch Logs format includes `schema_version_id` and `bucket_arn`, which are not present in the general purpose bucket text format.
- **No parsing required** – Because CloudWatch Logs logs are structured JSON, you can query individual fields directly in CloudWatch Logs Insights without regex parsing. The general purpose bucket text format requires a regex or custom parser to extract fields.

Managing log retention

When you deliver server access logs to CloudWatch Logs, log retention is controlled by the retention setting on the log group. You set retention when you create the log group, or update it at any time using the CloudWatch Logs console, AWS CLI, or API.

Retention options range from 1 day to 10 years, or you can choose to never expire logs. When log data reaches the end of its retention period, it is automatically deleted from the log group.

If you have enabled the S3 Tables mirror for your log group, the mirrored data in the S3 table follows the same retention. When log data expires from the log group, the corresponding records are removed from the S3 table. You do not need to manage retention separately for the S3 table.

To set or update log group retention

1. Open the CloudWatch Logs console at CloudWatch Logs console.
2. Select the log group that receives your server access logs.
3. Choose **Actions**, then **Edit retention setting**.
4. Select a retention period and choose **Save**.

To set retention using the AWS CLI:

```
aws logs put-retention-policy \
  --log-group-name "/aws/vendedlogs/s3/DOC-EXAMPLE-BUCKET/S3_SERVER_ACCESS_LOGS" \
  --retention-in-days 90
```

For more information, see Change log data retention in CloudWatch Logs in the *CloudWatch Logs User Guide*.

Querying logs with CloudWatch Logs Insights

CloudWatch Logs Insights lets you interactively search and analyze your server access log data. Queries run against the log group where your logs are delivered and return results in seconds.

> **Note**
>
> For querying logs delivered to an Amazon S3 general purpose bucket, see Querying access logs for requests by using Amazon Athena. For querying the S3 Tables mirror with SQL, see Querying access logs in S3 Tables.

Getting started

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.

2. In the navigation pane, choose **Logs**, then **Logs Insights**.
3. Select your server access log group from the log group selector.
4. Choose a time range for your query.
5. Enter a query and choose **Run query**.

Traffic analysis

The following queries help you understand request volume and traffic patterns.

Example Request volume over time

```
stats count(*) as requests by bin(5m) as interval
| sort interval asc
```

Example Request mix by operation type

```
stats count(*) as cnt by operation
| sort cnt desc
```

Example Traffic by bucket

```
stats count(*) as requests, sum(bytes_sent_size) as bytes_out by bucket_arn
| sort requests desc
```

Error troubleshooting

The following queries help you identify and diagnose errors.

Example Error rate breakdown

```
filter http_status >= 400
| stats count(*) as error_count by http_status, error_code, operation
| sort error_count desc
```

Example 403 Access Denied requests

```
filter http_status = 403
| stats count(*) as denied_count by key_name, remote_ip, requester
```

```
| sort denied_count desc
```

Example 404 Not Found requests

```
filter http_status = 404
| stats count(*) as miss_count by key_name, error_code, remote_ip
| sort miss_count desc
```

Example 503 SlowDown (throttling) events

```
filter http_status = 503
| stats count(*) as throttle_count by key_name, remote_ip
| sort throttle_count desc
```

Access patterns

The following queries help you understand who is accessing your data and how.

Example Traffic by source IP

```
stats count(*) as total_requests,
  sum(http_status >= 400) as errors,
  sum(bytes_sent_size) as bytes_transferred
  by remote_ip
| sort total_requests desc
```

Example Operations by requester

```
stats count(*) as requests by requester, operation
| sort requests desc
```

Example Most accessed keys

```
filter operation like /REST\.(GET|PUT)\.OBJECT/
| stats count(*) as access_count, sum(bytes_sent_size) as bytes_out by key_name
| sort access_count desc
| limit 10
```

Latency analysis

The following queries help you identify slow requests and latency trends.

Example Latency by operation type

```
filter operation like /REST\.(GET|PUT)\.OBJECT/
| stats avg(total_duration) as avg_ms,
  pct(total_duration, 50) as p50_ms,
  pct(total_duration, 95) as p95_ms,
  max(total_duration) as max_ms,
  count(*) as requests
  by operation
```

Example Slowest keys by p95 latency

```
filter operation like /REST\.GET\.OBJECT/
| stats avg(total_duration) as avg_ms,
  pct(total_duration, 95) as p95_ms,
  count(*) as requests
  by key_name
| sort p95_ms desc
| limit 10
```

Example Latency over time

```
filter operation like /REST\.(GET|PUT)\.OBJECT/
| stats avg(total_duration) as avg_ms, pct(total_duration, 95) as p95_ms by bin(5m) as
 interval
| sort interval asc
```

Cost attribution

The following queries help you understand data transfer and identify cost drivers.

Example Data transfer by operation

```
filter operation like /REST\.(GET|PUT|DELETE)\.OBJECT/
| stats sum(bytes_sent_size) as total_bytes_out,
  sum(object_size) as total_object_bytes,
  count(*) as request_count
  by operation
| sort total_bytes_out desc
```

Example Largest objects being served

```
filter operation like /REST\.GET\.OBJECT/ and bytes_sent_size > 0
| stats max(object_size) as max_size, avg(object_size) as avg_size, count(*) as reads
 by key_name
| sort max_size desc
| limit 10
```

Security analysis

The following queries help you detect suspicious access patterns.

Example IPs with high error rates

```
stats count(*) as total,
  sum(http_status >= 400) as errors
  by remote_ip
| filter errors > 10
| sort errors desc
```

Example Failed access attempts by key

```
filter http_status >= 400
| stats count(*) as attempts by remote_ip, key_name, operation, http_status
| sort attempts desc
| limit 20
```

Example TLS version distribution

```
stats count(*) as cnt by tls_version
| sort cnt desc
```

Querying access logs in S3 Tables

When you enable the Amazon S3 Tables integration, Amazon CloudWatch Logs delivers your server access logs in Apache Iceberg format to the `aws-cloudwatch` managed table bucket. Because the data is in Iceberg format, you can query it with any tool that supports Apache Iceberg, including:

- **Amazon Athena** – Run SQL queries directly from the Athena console using the S3 Tables catalog.
- **Amazon SageMaker Unified Studio** – Use the built-in SQL editor to query your log data alongside other analytics workloads.

- **Amazon Redshift** – Query Iceberg tables through Redshift Spectrum or direct integration.
- **Apache Spark on Amazon EMR** – Use Spark SQL or the DataFrame API with the Iceberg connector.
- **Open-source tools** – Any tool that supports the Apache Iceberg REST catalog, such as Trino, DuckDB, or PyIceberg.

> **Note**
>
> For querying logs interactively in CloudWatch Logs Insights (without SQL), see Querying logs with CloudWatch Logs Insights.

Prerequisites

Before you can query access logs in S3 Tables, you must complete the following:

1. **Enable the S3 Tables integration.** See Enabling the S3 Tables integration (optional).
2. **Enable S3 Tables integration with analytics services.** In the Amazon S3 console, navigate to **Table buckets** and choose **Enable integration**. This creates the `s3tablescatalog` federated catalog in Data Catalog and makes your tables visible to analytics services. By default, access is controlled through IAM permissions. You can optionally use AWS Lake Formation for fine-grained access control. For more information, see Integrating S3 Tables with AWS analytics services.
3. **Wait for data to populate.** Data typically appears within an hour of the first delivery to CloudWatch Logs.

Connecting Amazon Athena

1. Open the Athena console at https://console.aws.amazon.com/athena/.
2. In the query editor, select the **Amazon S3 Tables** catalog from the data source dropdown.
3. Your log tables appear in the database list under the `logs` namespace.

The table name for server access logs is `amazon_s3__server_access`. The fully qualified table reference is:

```
"s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
```

Traffic analysis

Example Request volume over time

```
SELECT date_trunc('minute', request_time) AS interval,
  COUNT(*) AS requests
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
GROUP BY date_trunc('minute', request_time)
ORDER BY interval ASC;
```

Example Request mix by operation type

```
SELECT operation, COUNT(*) AS cnt
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
GROUP BY operation
ORDER BY cnt DESC;
```

Example Traffic by bucket

```
SELECT bucket_arn, COUNT(*) AS requests, SUM(bytes_sent_size) AS bytes_out
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
GROUP BY bucket_arn
ORDER BY requests DESC;
```

Error troubleshooting

Example Error rate breakdown

```
SELECT http_status, error_code, operation, COUNT(*) AS error_count
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
WHERE http_status >= 400
GROUP BY http_status, error_code, operation
ORDER BY error_count DESC;
```

Example 403 Access Denied requests

```
SELECT key_name, remote_ip, requester, COUNT(*) AS denied_count
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
WHERE http_status = 403
```

```
GROUP BY key_name, remote_ip, requester
ORDER BY denied_count DESC;
```

Example 404 Not Found requests

```
SELECT key_name, error_code, remote_ip, COUNT(*) AS miss_count
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
WHERE http_status = 404
GROUP BY key_name, error_code, remote_ip
ORDER BY miss_count DESC;
```

Example 503 SlowDown (throttling) events

```
SELECT key_name, remote_ip, COUNT(*) AS throttle_count
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
WHERE http_status = 503
GROUP BY key_name, remote_ip
ORDER BY throttle_count DESC;
```

Access patterns

Example Traffic by source IP

```
SELECT remote_ip,
  COUNT(*) AS total_requests,
  SUM(CASE WHEN http_status >= 400 THEN 1 ELSE 0 END) AS errors,
  SUM(bytes_sent_size) AS bytes_transferred
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
GROUP BY remote_ip
ORDER BY total_requests DESC;
```

Example Operations by requester

```
SELECT requester, operation, COUNT(*) AS requests
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
GROUP BY requester, operation
ORDER BY requests DESC;
```

Example Most accessed keys

```
SELECT key_name, COUNT(*) AS access_count, SUM(bytes_sent_size) AS bytes_out
```

```
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
WHERE operation IN ('REST.GET.OBJECT', 'REST.PUT.OBJECT')
GROUP BY key_name
ORDER BY access_count DESC
LIMIT 10;
```

Latency analysis

Example Latency by operation type

```
SELECT operation,
  AVG(total_duration) AS avg_ms,
  APPROX_PERCENTILE(total_duration, 0.5) AS p50_ms,
  APPROX_PERCENTILE(total_duration, 0.95) AS p95_ms,
  MAX(total_duration) AS max_ms,
  COUNT(*) AS requests
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
WHERE operation IN ('REST.GET.OBJECT', 'REST.PUT.OBJECT')
GROUP BY operation;
```

Example Slowest keys by p95 latency

```
SELECT key_name,
  AVG(total_duration) AS avg_ms,
  APPROX_PERCENTILE(total_duration, 0.95) AS p95_ms,
  COUNT(*) AS requests
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
WHERE operation = 'REST.GET.OBJECT'
GROUP BY key_name
ORDER BY p95_ms DESC
LIMIT 10;
```

Example Latency over time

```
SELECT date_trunc('minute', request_time) AS interval,
  AVG(total_duration) AS avg_ms,
  APPROX_PERCENTILE(total_duration, 0.95) AS p95_ms
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
WHERE operation IN ('REST.GET.OBJECT', 'REST.PUT.OBJECT')
GROUP BY date_trunc('minute', request_time)
ORDER BY interval ASC;
```

Cost attribution

Example Data transfer by operation

```
SELECT operation,
  SUM(bytes_sent_size) AS total_bytes_out,
  SUM(object_size) AS total_object_bytes,
  COUNT(*) AS request_count
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
WHERE operation IN ('REST.GET.OBJECT', 'REST.PUT.OBJECT', 'REST.DELETE.OBJECT')
GROUP BY operation
ORDER BY total_bytes_out DESC;
```

Example Largest objects being served

```
SELECT key_name, MAX(object_size) AS max_size, AVG(object_size) AS avg_size, COUNT(*)
 AS reads
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
WHERE operation = 'REST.GET.OBJECT' AND bytes_sent_size > 0
GROUP BY key_name
ORDER BY max_size DESC
LIMIT 10;
```

Security analysis

Example IPs with high error rates

```
SELECT remote_ip,
  COUNT(*) AS total,
  SUM(CASE WHEN http_status >= 400 THEN 1 ELSE 0 END) AS errors
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
GROUP BY remote_ip
HAVING SUM(CASE WHEN http_status >= 400 THEN 1 ELSE 0 END) > 10
ORDER BY errors DESC;
```

Example Failed access attempts by key

```
SELECT remote_ip, key_name, operation, http_status, COUNT(*) AS attempts
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
WHERE http_status >= 400
GROUP BY remote_ip, key_name, operation, http_status
ORDER BY attempts DESC
```

```
LIMIT 20;
```

Example TLS version distribution

```
SELECT tls_version, COUNT(*) AS cnt
FROM "s3tablescatalog/aws-cloudwatch"."logs"."amazon_s3__server_access"
GROUP BY tls_version
ORDER BY cnt DESC;
```

Using AI agents to query Amazon S3 server access logs

MCP-compatible AI agents can discover the S3 tables that contain your Amazon S3 server access logs, generate SQL queries, and analyze the results. Because your Amazon S3 server access logs are exported as Iceberg tables in the `aws-cloudwatch` table bucket, agents can query them through the same S3 Tables catalog that you use with Amazon Athena, Amazon EMR, Amazon Redshift, or any Iceberg-compatible tool.

The Agent Toolkit for AWS provides skills that give agents validated procedures for working with AWS data. The Querying CloudWatch Logs system tables skill runs SQL queries on CloudWatch Logs data exported to S3 Tables, including your Amazon S3 server access logs in the `aws-cloudwatch` table bucket. To use it, download the skill from the repository on GitHub and add it to your agent's skills directory.

For example, you can ask an agent to show the top 10 source IP addresses by request count in your S3 server access logs over the last 24 hours. The agent uses the skill to locate the access log table and run the query in Athena.

Troubleshooting CloudWatch Logs delivery

Logs not appearing in CloudWatch Logs log group

If you have configured delivery but do not see logs in your CloudWatch Logs log group, check the following:

- **Verify the delivery is active.** Run `aws logs describe-deliveries` and confirm that your delivery shows a status of `ACTIVE`.
- **Verify the delivery source.** Run `aws logs describe-delivery-sources` and confirm that the source bucket ARN and log type (`S3_SERVER_ACCESS_LOGS`) are correct.
- **Confirm the source bucket has traffic.** Server access logs are only generated when requests are made to the source bucket. If the bucket has no traffic, no logs are produced.

- **Check the log group Region.** The log group must be in the same AWS Region as the source bucket.
- **Allow time for initial delivery.** After setup, it may take a few minutes for the first logs to appear.

S3 Tables integration not populating

If logs appear in CloudWatch Logs but the S3 Tables integration is empty, check the following:

- **Verify the integration exists.** In the CloudWatch console, go to **Settings**, **Global**, and confirm that an S3 Table Integration is listed.
- **Verify the source is associated.** Run `aws logs list-sources-for-s3-table-integration --integration-arn` *`integration-arn`* and confirm that your delivery source name and type appear in the list.
- **Check the service role permissions.** The IAM role used for the integration must have the `logs:integrateWithS3Table` permission scoped to your log group ARN. See Enabling the S3 Tables integration (optional).
- **Confirm the `aws-cloudwatch` table bucket exists.** In the Amazon S3 console, navigate to **Table buckets** and look for the `aws-cloudwatch` bucket. If it does not exist, the integration may not have completed successfully.
- **Allow time for data to appear.** The integration may lag behind CloudWatch Logs delivery. Data typically appears within an hour of the first delivery to CloudWatch Logs.
- **Check that the integration processes only new data.** The S3 Tables integration does not backfill logs from before the association was created. Only log events added after the association are delivered to S3 Tables.

Permission errors

The following are common permission errors and how to resolve them.

- **Access Denied on `PutDeliverySource`** – The caller needs the `logs:PutDeliverySource` and `s3:AllowVendedLogDeliveryForResource` permission.
- **Access Denied on `CreateDelivery`** – The caller needs the `logs:CreateDelivery` permission.
- **Access Denied on `CreateS3TableIntegration`** – The caller needs the `observabilityadmin:CreateS3TableIntegration` permission, plus `s3tables:CreateTableBucket`, `s3tables:PutTableBucketEncryption`, and `s3tables:PutTableBucketPolicy`.

- **AWS KMS errors on log group** – If your log group uses AWS KMS encryption, verify that the AWS KMS key policy grants access to the CloudWatch Logs service principal. See Encryption.
- **AWS KMS errors on S3 Tables integration** – If your log group uses AWS KMS encryption and the S3 Tables integration is enabled, verify that the AWS KMS key policy grants access to both `systemtables.cloudwatch.amazonaws.com` and `maintenance.s3tables.amazonaws.com`. See Encryption.
- **Access Denied when querying in Athena** – Verify that the IAM principal running the query has permissions to access the `aws-cloudwatch` table bucket and the S3 Tables catalog. If you are using AWS Lake Formation, verify that `SELECT` and `DESCRIBE` grants are in place.

Delivery stopped unexpectedly

If logs were previously being delivered but have stopped, check the following:

- **Source bucket deleted.** If the source bucket was deleted, delivery stops. If the bucket is recreated, you must create a new delivery again.
- **Log group deleted.** If the log group was deleted, delivery stops. Create a new log group and update the delivery destination.
- **IAM permissions revoked.** If the permissions required for delivery were removed, delivery may stop. Verify that the delivery configuration still has the required permissions.
- **Service quotas.** Check CloudWatch Logs service quotas for delivery limits per account. You may have reached the maximum number of deliveries.

Managing ingestion costs

If your CloudWatch Logs ingestion costs are higher than expected, consider the following:

- **Reduce retention.** Set a shorter retention period on the log group to reduce storage costs. CloudWatch Logs automatically deletes data that exceeds the retention period.
- **Review source bucket traffic.** High-traffic buckets generate more logs. Consider whether all source buckets need CloudWatch Logs delivery, or whether the free general purpose bucket delivery path is sufficient for some buckets.
- **Use volume-based pricing.** CloudWatch Logs vended logs ingestion pricing is tiered by volume. Higher volumes receive lower per-GB rates. For current rates, see CloudWatch pricing.

Delivering logs to an Amazon S3 general purpose bucket

When you deliver server access logs to an Amazon S3 general purpose bucket, logs are written in space-delimited text format. There is no charge for log delivery. The destination bucket must be in the same AWS Region and AWS account as the source bucket.

Topics

- Enabling Amazon S3 server access logging
- Amazon S3 server access log format
- Managing log retention
- Using Amazon S3 server access logs to identify requests
- Troubleshoot server access logging

Enabling Amazon S3 server access logging

Server access logging provides detailed records for the requests that are made to an Amazon S3 bucket. Server access logs are useful for many applications. For example, access log information can be useful in security and access audits. This information can also help you learn about your customer base and understand your Amazon S3 bill.

> **Note**
>
> This page covers delivering server access logs to an Amazon S3 general purpose bucket. You can also deliver logs to Amazon CloudWatch Logs for structured querying, cross-account aggregation, and optional mirroring to S3 Tables. For more information, see Delivering server access logs to CloudWatch Logs.

By default, Amazon S3 doesn't collect server access logs. When you enable logging, Amazon S3 delivers access logs for a source bucket to a destination bucket (also known as a *target bucket*) that you choose. The destination bucket must be in the same AWS Region and AWS account as the source bucket.

An access log record contains details about the requests that are made to a bucket. This information can include the request type, the resources that are specified in the request, and the time and date that the request was processed. For more information about logging basics, see Logging requests with server access logging.

Important

- There is no extra charge for enabling server access logging on an Amazon S3 bucket. However, any log files that the system delivers to you will accrue the usual charges for storage. (You can delete the log files at any time.) We do not assess data-transfer charges for log file delivery, but we do charge the normal data-transfer rate for accessing the log files.
- Your destination bucket should not have server access logging enabled. You can have logs delivered to any bucket that you own that is in the same Region as the source bucket, including the source bucket itself. However, delivering logs to the source bucket will cause an infinite loop of logs and is not recommended. For simpler log management, we recommend that you save access logs in a different bucket.
- S3 buckets that have S3 Object Lock enabled can't be used as destination buckets for server access logs. Your destination bucket must not have a default retention period configuration.
- The destination bucket must not have Requester Pays enabled.

You can enable or disable server access logging by using the Amazon S3 console, Amazon S3 API, the AWS Command Line Interface (AWS CLI), or AWS SDKs.

Permissions for log delivery

Amazon S3 uses a special log delivery account to write server access logs. These writes are subject to the usual access control restrictions. For access log delivery, you must grant the logging service principal (`logging.s3.amazonaws.com`) access to your destination bucket.

To grant permissions to Amazon S3 for log delivery, you can use either a bucket policy or bucket access control lists (ACLs), depending on your destination bucket's S3 Object Ownership settings. However, we recommend that you use a bucket policy instead of ACLs.

Note

Granting `s3:PutObject` to the logging service principal is not sufficient if the destination bucket uses SSE-KMS default encryption. The destination bucket must use Amazon S3 managed keys (SSE-S3). If the destination bucket uses SSE-KMS, Amazon S3 might deliver log objects that are encrypted with a key that you can't access.

Bucket owner enforced setting for S3 Object Ownership

If the destination bucket uses the Bucket owner enforced setting for Object Ownership, ACLs are disabled and no longer affect permissions. In this case, you must update the bucket policy for the destination bucket to grant access to the logging service principal. You can't update your bucket ACL to grant access to the S3 log delivery group. You also can't include destination grants (also known as *target grants*) in your PutBucketLogging configuration.

For information about migrating existing bucket ACLs for access log delivery to a bucket policy, see Grant access to the S3 log delivery group for server access logging. For more information about Object Ownership, see Controlling ownership of objects and disabling ACLs for your bucket. When you create new buckets, ACLs are disabled by default.

Granting access by using a bucket policy

To grant access by using the bucket policy on the destination bucket, update the bucket policy to grant the `s3:PutObject` permission to the logging service principal. If you use the Amazon S3 console to enable server access logging, the console automatically updates the bucket policy on the destination bucket to grant this permission to the logging service principal. If you enable server access logging programmatically, you must manually update the bucket policy for the destination bucket to grant access to the logging service principal.

For an example bucket policy that grants access to the logging service principal, see the section called “Grant permissions to the logging service principal by using a bucket policy”.

Granting access by using bucket ACLs

You can alternately use bucket ACLs to grant access for access log delivery. You add a grant entry to the bucket ACL that grants `WRITE` and `READ_ACP` permissions to the S3 log delivery group. However, granting access to the S3 log delivery group by using bucket ACLs is not recommended. For more information, see Controlling ownership of objects and disabling ACLs for your bucket. For information about migrating existing bucket ACLs for access log delivery to a bucket policy, see Grant access to the S3 log delivery group for server access logging. For an example ACL that grants access to the logging service principal, see the section called “Grant permissions to the log delivery group by using a bucket ACL”.

Grant permissions to the logging service principal by using a bucket policy

This example bucket policy grants the `s3:PutObject` permission to the logging service principal (`logging.s3.amazonaws.com`). To use this bucket policy, replace the *`user`*

input placeholders with your own information. In the following policy, *amzn-s3-demo-destination-bucket* is the destination bucket where server access logs will be delivered, and *amzn-s3-demo-source-bucket* is the source bucket. *EXAMPLE-LOGGING-PREFIX* is the optional destination prefix (also known as a *target prefix*) that you want to use for your log objects. *SOURCE-ACCOUNT-ID* is the AWS account that owns the source bucket.

> **Note**
>
> If there are Deny statements in your bucket policy, make sure that they don't prevent Amazon S3 from delivering access logs.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "S3ServerAccessLogsPolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "logging.s3.amazonaws.com"
            },
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket/EXAMPLE-LOGGING-PREFIX*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

Grant permissions to the log delivery group by using a bucket ACL

> **Note**
>
> As a security best practice, Amazon S3 disables access control lists (ACLs) by default in all new buckets. For more information about ACL permissions in the Amazon S3 console, see Configuring ACLs.

Although we do not recommend this approach, you can grant permissions to the log delivery group by using a bucket ACL. However, if the destination bucket uses the Bucket owner enforced setting for Object Ownership, you can't set bucket or object ACLs. You also can't include destination grants (also known as *target grants*) in your PutBucketLogging configuration. Instead, you must use a bucket policy to grant access to the logging service principal (`logging.s3.amazonaws.com`). For more information, see Permissions for log delivery.

In the bucket ACL, the log delivery group is represented by the following URL:

```
http://acs.amazonaws.com/groups/s3/LogDelivery
```

To grant `WRITE` and `READ_ACP` (ACL read) permissions, add the following grants to the destination bucket ACL:

```
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>WRITE</Permission>
</Grant>
<Grant>
    <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/s3/LogDelivery</URI>
    </Grantee>
    <Permission>READ_ACP</Permission>
</Grant>
```

For examples of adding ACL grants programmatically, see Configuring ACLs.

Important

When you enable Amazon S3 server access logging by using AWS CloudFormation on a bucket and you're using ACLs to grant access to the S3 log delivery group, you must also add `"AccessControl": "LogDeliveryWrite"` to your CloudFormation template. Doing so is important because you can grant those permissions only by creating an ACL for the bucket, but you can't create custom ACLs for buckets in CloudFormation. You can use only canned ACLs with CloudFormation.

To enable server access logging

To enable server access logging by using the Amazon S3 console, Amazon S3 REST API, AWS SDKs, and AWS CLI, use the following procedures.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to enable server access logging for.
4. Choose **Properties**.
5. In the **Server access logging** section, choose **Edit**.
6. Under **Server access logging**, choose **Enable**.
7. Under **Destination bucket**, specify a bucket and an optional prefix. If you specify a prefix, we recommend including a forward slash (/) after the prefix to make it easier to find your logs.

Note

Specifying a prefix with a slash (/) makes it simpler for you to locate the log objects. For example, if you specify the prefix value `logs/`, each log object that Amazon S3 creates begins with the `logs/` prefix in its key, as follows:

```
logs/2013-11-01-21-32-16-E568B2907131C0C0
```

If you specify the prefix value `logs`, the log object appears as follows:

```
logs2013-11-01-21-32-16-E568B2907131C0C0
```

8. Under **Log object key format**, do one of the following:

   - To choose non-date-based partitioning, choose **[DestinationPrefix][YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]**.
   - To choose date-based partitioning, choose **[DestinationPrefix][SourceAccountId]/[SourceRegion]/[SourceBucket]/[YYYY]/[MM]/[DD]/[YYYY]-[MM]-[DD]-[hh]-[mm]-[ss]-[UniqueString]**, then choose **S3 event time** or **Log file delivery time**.

9. Choose **Save changes**.

   When you enable server access logging on a bucket, the console both enables logging on the source bucket and updates the bucket policy for the destination bucket to grant the `s3:PutObject` permission to the logging service principal (`logging.s3.amazonaws.com`). For more information about this bucket policy, see Grant permissions to the logging service principal by using a bucket policy.

   You can view the logs in the destination bucket. After you enable server access logging, it might take a few hours before the logs are delivered to the target bucket.

For more information, see Viewing the properties for an S3 general purpose bucket.

Using the REST API

To enable logging, you submit a PutBucketLogging request to add the logging configuration on the source bucket. The request specifies the destination bucket (also known as a *target bucket*) and, optionally, the prefix to be used with all log object keys.

The following example identifies *amzn-s3-demo-destination-bucket* as the destination bucket and *logs/* as the prefix.

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>amzn-s3-demo-destination-bucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
  </LoggingEnabled>
</BucketLoggingStatus>
```

The following example identifies *amzn-s3-demo-destination-bucket* as the destination bucket, *logs/* as the prefix, and `EventTime` as the log object key format.

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <LoggingEnabled>
    <TargetBucket>amzn-s3-demo-destination-bucket</TargetBucket>
    <TargetPrefix>logs/</TargetPrefix>
    <TargetObjectKeyFormat>
      <PartitionedPrefix>
         <PartitionDateSource>EventTime</PartitionDateSource>
      </PartitionedPrefix>
  </TargetObjectKeyFormat>
  </LoggingEnabled>
</BucketLoggingStatus>
```

The log objects are written and owned by the S3 log delivery account, and the bucket owner is granted full permissions on the log objects. You can optionally use destination grants (also known as *target grants*) to grant permissions to other users so that they can access the logs. For more information, see PutBucketLogging.

> **Note**
>
> If the destination bucket uses the Bucket owner enforced setting for Object Ownership, you can't use destination grants to grant permissions to other users. To grant permissions to others, you can update the bucket policy on the destination bucket. For more information, see Permissions for log delivery.

To retrieve the logging configuration on a bucket, use the GetBucketLogging API operation.

To delete the logging configuration, you send a `PutBucketLogging` request with an empty `BucketLoggingStatus`:

```
<BucketLoggingStatus xmlns="http://doc.s3.amazonaws.com/2006-03-01">
</BucketLoggingStatus>
```

To enable logging on a bucket, you can use either the Amazon S3 API or the AWS SDK wrapper libraries.

Using the AWS SDKs

The following examples enable logging on a bucket. You must create two buckets, a source bucket and a destination (target) bucket. The examples update the bucket ACL on the destination bucket first. They then grant the log delivery group the necessary permissions to write logs to the destination bucket, and then they enable logging on the source bucket.

These examples won't work on destination buckets that use the Bucket owner enforced setting for Object Ownership.

If the destination (target) bucket uses the Bucket owner enforced setting for Object Ownership, you can't set bucket or object ACLs. You also can't include destination (target) grants in your PutBucketLogging configuration. You must use a bucket policy to grant access to the logging service principal (`logging.s3.amazonaws.com`). For more information, see Permissions for log delivery.

.NET

SDK for .NET

> **Note**
>
> There's more on GitHub. Find the complete example and learn how to set up and run in the AWS Code Examples Repository.

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
/// </summary>
public class ServerAccessLogging
```

```
    {
        private static IConfiguration _configuration = null!;

        public static async Task Main()
        {
            LoadConfig();

            string bucketName = _configuration["BucketName"];
            string logBucketName = _configuration["LogBucketName"];
            string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
            string accountId = _configuration["AccountId"];

            // If the AWS Region defined for your default user is different
            // from the Region where your Amazon S3 bucket is located,
            // pass the Region name to the Amazon S3 client object's constructor.
            // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
            IAmazonS3 client = new AmazonS3Client();

            try
            {
                // Update bucket policy for target bucket to allow delivery of
logs to it.
                await SetBucketPolicyToAllowLogDelivery(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix,
                    accountId);

                // Enable logging on the source bucket.
                await EnableLoggingAsync(
                    client,
                    bucketName,
                    logBucketName,
                    logObjectKeyPrefix);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine($"Error: {e.Message}");
            }
        }

        /// <summary>
        /// This method grants appropriate permissions for logging to the
```

```
        /// Amazon S3 bucket where the logs will be stored.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
used
        /// to apply the bucket policy.</param>
        /// <param name="sourceBucketName">The name of the source bucket.</param>
        /// <param name="logBucketName">The name of the bucket where logging
        /// information will be stored.</param>
        /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
        /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
        /// <returns>Async task.</returns>
        public static async Task SetBucketPolicyToAllowLogDelivery(
            IAmazonS3 client,
            string sourceBucketName,
            string logBucketName,
            string logPrefix,
            string accountId)
        {
            var resourceArn = @"""arn:aws:s3:::" + logBucketName + "/" +
logPrefix + @"*""";

            var newPolicy = @"{
                                ""Statement"":[{
                                ""Sid"": ""S3ServerAccessLogsPolicy"",
                                ""Effect"": ""Allow"",
                                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                                ""Action"": [""s3:PutObject""],
                                ""Resource"": [" + resourceArn + @"],
                                ""Condition"": {
                                ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3:::" + sourceBucketName + @""" },
                                ""StringEquals"": { ""aws:SourceAccount"": """ +
accountId + @""" }
                                        }
                                    }]
                                }";
            Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging:");
            Console.WriteLine(newPolicy);

            PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
```

```
            {
                BucketName = logBucketName,
                Policy = newPolicy,
            };
            await client.PutBucketPolicyAsync(putRequest);
            Console.WriteLine("Policy applied.");
        }

        /// <summary>
        /// This method enables logging for an Amazon S3 bucket. Logs will be
 stored
        /// in the bucket you selected for logging. Selected prefix
        /// will be prepended to each log object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 client which will be
 used
        /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
        /// <param name="bucketName">The name of the Amazon S3 bucket for which
 you
        /// wish to enable logging.</param>
        /// <param name="logBucketName">The name of the Amazon S3 bucket where
 logging
        /// information will be stored.</param>
        /// <param name="logObjectKeyPrefix">The prefix to prepend to each
        /// object key.</param>
        /// <returns>Async task.</returns>
        public static async Task EnableLoggingAsync(
            IAmazonS3 client,
            string bucketName,
            string logBucketName,
            string logObjectKeyPrefix)
        {
            Console.WriteLine($"Enabling logging for bucket {bucketName}.");
            var loggingConfig = new S3BucketLoggingConfig
            {
                TargetBucketName = logBucketName,
                TargetPrefix = logObjectKeyPrefix,
            };

            var putBucketLoggingRequest = new PutBucketLoggingRequest
            {
                BucketName = bucketName,
                LoggingConfig = loggingConfig,
```

```
            };
            await client.PutBucketLoggingAsync(putBucketLoggingRequest);
            Console.WriteLine($"Logging enabled.");
        }

        /// <summary>
        /// Loads configuration from settings files.
        /// </summary>
        public static void LoadConfig()
        {
            _configuration = new ConfigurationBuilder()
                .SetBasePath(Directory.GetCurrentDirectory())
                .AddJsonFile("settings.json") // Load settings from .json file.
                .AddJsonFile("settings.local.json", true) // Optionally, load
  local settings.
                .Build();
        }
    }
```

- For API details, see PutBucketLogging in *AWS SDK for .NET API Reference*.

Java

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLoggingStatus;
import software.amazon.awssdk.services.s3.model.LoggingEnabled;
import software.amazon.awssdk.services.s3.model.PartitionedPrefix;
import software.amazon.awssdk.services.s3.model.PutBucketLoggingRequest;
import software.amazon.awssdk.services.s3.model.TargetObjectKeyFormat;

// Class to set a bucket policy on a target S3 bucket and enable server access
 logging on a source S3 bucket.
public class ServerAccessLogging {
    private static S3Client s3Client;

    public static void main(String[] args) {
        String sourceBucketName = "SOURCE-BUCKET";
        String targetBucketName = "TARGET-BUCKET";
        String sourceAccountId = "123456789012";
        String targetPrefix = "logs/";
```

```
        // Create S3 Client.
        s3Client = S3Client.builder().
                region(Region.US_EAST_2)
                .build();

        // Set a bucket policy on the target S3 bucket to enable server access
 logging by granting the
        // logging.s3.amazonaws.com principal permission to use the PutObject
 operation.
        ServerAccessLogging serverAccessLogging = new ServerAccessLogging();
        serverAccessLogging.setTargetBucketPolicy(sourceAccountId, sourceBucketName,
 targetBucketName);

        // Enable server access logging on the source S3 bucket.
        serverAccessLogging.enableServerAccessLogging(sourceBucketName,
 targetBucketName,
                targetPrefix);

    }

    // Function to set a bucket policy on the target S3 bucket to enable server
 access logging by granting the
    // logging.s3.amazonaws.com principal permission to use the PutObject operation.
    public void setTargetBucketPolicy(String sourceAccountId, String
 sourceBucketName, String targetBucketName) {
        String policy = "{\n" +
                "    \"Version\": \"2012-10-17\",\n" +
                "    \"Statement\": [\n" +
                "        {\n" +
                "            \"Sid\": \"S3ServerAccessLogsPolicy\",\n" +
                "            \"Effect\": \"Allow\",\n" +
                "            \"Principal\": {\"Service\": \"logging.s3.amazonaws.com
\"},\n" +
                "            \"Action\": [\n" +
                "                \"s3:PutObject\"\n" +
                "            ],\n" +
                "            \"Resource\": \"arn:aws:s3:::" + targetBucketName + "/*
\",\n" +
                "            \"Condition\": {\n" +
                "                \"ArnLike\": {\n" +
                "                    \"aws:SourceArn\": \"arn:aws:s3:::" +
 sourceBucketName + "\"\n" +
                "                },\n" +
```

```
                "                \"StringEquals\": {\n" +
                "                    \"aws:SourceAccount\": \"" + sourceAccountId +
 "\"\n" +
                "                }\n" +
                "            }\n" +
                "        }\n" +
                "    ]\n" +
                "}";
        s3Client.putBucketPolicy(b -> b.bucket(targetBucketName).policy(policy));
    }

    // Function to enable server access logging on the source S3 bucket.
    public void enableServerAccessLogging(String sourceBucketName, String
 targetBucketName,
            String targetPrefix) {
        TargetObjectKeyFormat targetObjectKeyFormat =
 TargetObjectKeyFormat.builder()

 .partitionedPrefix(PartitionedPrefix.builder().partitionDateSource("EventTime").build())
                .build();
        LoggingEnabled loggingEnabled = LoggingEnabled.builder()
                .targetBucket(targetBucketName)
                .targetPrefix(targetPrefix)
                .targetObjectKeyFormat(targetObjectKeyFormat)
                .build();
        BucketLoggingStatus bucketLoggingStatus = BucketLoggingStatus.builder()
                .loggingEnabled(loggingEnabled)
                .build();
        s3Client.putBucketLogging(PutBucketLoggingRequest.builder()
                .bucket(sourceBucketName)
                .bucketLoggingStatus(bucketLoggingStatus)
                .build());
    }

}
```

Using the AWS CLI

We recommend that you create a dedicated logging bucket in each AWS Region that you have S3 buckets in. Then have your Amazon S3 access logs delivered to that S3 bucket. For more information and examples, see put-bucket-logging in the *AWS CLI Reference*.

If the destination (target) bucket uses the Bucket owner enforced setting for Object Ownership, you can't set bucket or object ACLs. You also can't include destination (target) grants in your PutBucketLogging configuration. You must use a bucket policy to grant access to the logging service principal (`logging.s3.amazonaws.com`). For more information, see Permissions for log delivery.

Example— Enable access logs with five buckets across two Regions

In this example, you have the following five buckets:

- *amzn-s3-demo-source-bucket*-us-east-1
- *amzn-s3-demo-source-bucket1*-us-east-1
- *amzn-s3-demo-source-bucket2*-us-east-1
- *amzn-s3-demo-bucket1*-us-west-2
- *amzn-s3-demo-bucket2*-us-west-2

> **Note**
>
> The final step of the following procedure provides example bash scripts that you can use to create your logging buckets and enable server access logging on these buckets. To use those scripts, you must create the `policy.json` and `logging.json` files, as described in the following procedure.

1. Create two logging destination buckets in the US West (Oregon) and US East (N. Virginia) Regions and give them the following names:
   - *amzn-s3-demo-destination-bucket*-logs-us-east-1
   - *amzn-s3-demo-destination-bucket1*-logs-us-west-2
2. Later in these steps, you will enable server access logging as follows:
   - *amzn-s3-demo-source-bucket*-us-east-1 logs to the S3 bucket *amzn-s3-demo-destination-bucket*-logs-us-east-1 with the prefix *amzn-s3-demo-source-bucket*-us-east-1
   - *amzn-s3-demo-source-bucket1*-us-east-1 logs to the S3 bucket *amzn-s3-demo-destination-bucket*-logs-us-east-1 with the prefix *amzn-s3-demo-source-bucket1*-us-east-1

- *amzn-s3-demo-source-bucket2*-us-east-1 logs to the S3 bucket *amzn-s3-demo-destination-bucket*-logs-us-east-1 with the prefix *amzn-s3-demo-source-bucket2*-us-east-1
- *amzn-s3-demo-bucket1*-us-west-2 logs to the S3 bucket *amzn-s3-demo-destination-bucket1*-logs-us-west-2 with the prefix *amzn-s3-demo-bucket1*-us-west-2
- *amzn-s3-demo-bucket2*-us-west-2 logs to the S3 bucket *amzn-s3-demo-destination-bucket1*-logs-us-west-2 with the prefix *amzn-s3-demo-bucket2*-us-west-2

3. For each destination logging bucket, grant permissions for server access log delivery by using a bucket ACL *or* a bucket policy:

   - **Update the bucket policy** (Recommended) – To grant permissions to the logging service principal, use the following `put-bucket-policy` command. Replace *amzn-s3-demo-destination-bucket*-logs with the name of your destination bucket.

     ```
     aws s3api put-bucket-policy --bucket amzn-s3-demo-destination-bucket-logs --
     policy file://policy.json
     ```

     `Policy.json` is a JSON document in the current folder that contains the following bucket policy. To use this bucket policy, replace the *user input placeholders* with your own information. In the following policy, *amzn-s3-demo-destination-bucket-logs* is the destination bucket where server access logs will be delivered, and *amzn-s3-demo-source-bucket* is the source bucket. *SOURCE-ACCOUNT-ID* is the AWS account that owns the source bucket.

     JSON

     ```
     {
         "Version":"2012-10-17",
         "Statement": [
             {
                 "Sid": "S3ServerAccessLogsPolicy",
                 "Effect": "Allow",
                 "Principal": {
                     "Service": "logging.s3.amazonaws.com"
                 },
     ```

```
            "Action": [
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::amzn-s3-demo-destination-bucket-logs/
*",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-source-
bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "SOURCE-ACCOUNT-ID"
                }
            }
        }
    ]
}
```

- **Update the bucket ACL** – To grant permissions to the S3 log delivery group, use the following `put-bucket-acl` command. Replace *amzn-s3-demo-destination-bucket-logs* with the name of your destination (target) bucket.

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-destination-bucket-logs  --
grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp
 URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

4. Then, create a `logging.json` file that contains your logging configuration (based on one of the three examples that follow). After you create the `logging.json` file, you can apply the logging configuration by using the following `put-bucket-logging` command. Replace *amzn-s3-demo-destination-bucket-logs* with the name of your destination (target) bucket.

```
aws s3api put-bucket-logging --bucket amzn-s3-demo-destination-bucket-logs --
bucket-logging-status file://logging.json
```

> **Note**
>
> Instead of using this `put-bucket-logging` command to apply the logging configuration on each destination bucket, you can use one of the bash scripts provided in the next step. To use those scripts, you must create the `policy.json` and `logging.json` files, as described in this procedure.

The `logging.json` file is a JSON document in the current folder that contains your logging configuration. If a destination bucket uses the Bucket owner enforced setting for Object Ownership, your logging configuration can't contain destination (target) grants. For more information, see Permissions for log delivery.

Example– logging.json without destination (target) grants

The following example `logging.json` file doesn't contain destination (target) grants. Therefore, you can apply this configuration to a destination (target) bucket that uses the Bucket owner enforced setting for Object Ownership.

```
{
    "LoggingEnabled": {
        "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
        "TargetPrefix": "amzn-s3-demo-destination-bucket/"
     }
 }
```

Example– logging.json with destination (target) grants

The following example `logging.json` file contains destination (target) grants.

If the destination bucket uses the Bucket owner enforced setting for Object Ownership, you can't include destination (target) grants in your PutBucketLogging configuration. For more information, see Permissions for log delivery.

```
{
    "LoggingEnabled": {
```

```
        "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
        "TargetPrefix": "amzn-s3-demo-destination-bucket/",
        "TargetGrants": [
             {
                "Grantee": {
                    "Type": "CanonicalUser",
                    "ID":
"79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be"
                 },
                "Permission": "FULL_CONTROL"
             }
         ]
     }
  }
```

Grantee values

You can specify the person (grantee) to whom you're assigning access rights (by using request elements) in the following ways:

- By the person's ID:

```
{
  "Grantee": {
    "Type": "CanonicalUser",
    "ID": "ID"
  }
}
```

- By URI:

```
{
  "Grantee": {
    "Type": "Group",
    "URI": "http://acs.amazonaws.com/groups/global/AuthenticatedUsers"
  }
}
```

Example– logging.json with the log object key format set to S3 event time

The following `logging.json` file changes the log object key format to S3 event time.

```
  {
    "LoggingEnabled": {
        "TargetBucket": "amzn-s3-demo-destination-bucket-logs",
        "TargetPrefix": "amzn-s3-demo-destination-bucket/",
        "TargetObjectKeyFormat": {
            "PartitionedPrefix": {
                "PartitionDateSource": "EventTime"
            }
         }
    }
}
```

5. Use one of the following bash scripts to add access logging for all the buckets in your account. Replace *amzn-s3-demo-destination-bucket-logs* with the name of your destination (target) bucket, and replace *us-west-2* with the name of the Region that your buckets are located in.

> **Note**
>
> This script works only if all of your buckets are in the same Region. If you have buckets in multiple Regions, you must adjust the script.

Example– Grant access with bucket policies and add logging for the buckets in your account

```
loggingBucket='amzn-s3-demo-destination-bucket-logs'
region='us-west-2'


# Create the logging bucket.
aws s3 mb s3://$loggingBucket --region $region

aws s3api put-bucket-policy --bucket $loggingBucket --policy file://policy.json
```

```
  # List the buckets in this account.
  buckets="$(aws s3 ls | awk '{print $3}')"

  # Put a bucket logging configuration on each bucket.
  for bucket in $buckets
      do
        # This if statement excludes the logging bucket.
        if [ "$bucket" == "$loggingBucket" ] ; then
            continue;
        fi
        printf '{
          "LoggingEnabled": {
            "TargetBucket": "%s",
            "TargetPrefix": "%s/"
        }
      }' "$loggingBucket" "$bucket"  > logging.json
      aws s3api put-bucket-logging --bucket $bucket --bucket-logging-status file://
logging.json
      echo "$bucket done"
  done

  rm logging.json

  echo "Complete"
```

Example– Grant access with bucket ACLs and add logging for the buckets in your account

```
  loggingBucket='amzn-s3-demo-destination-bucket-logs'
  region='us-west-2'


  # Create the logging bucket.
  aws s3 mb s3://$loggingBucket --region $region

  aws s3api put-bucket-acl --bucket $loggingBucket --grant-write URI=http://
acs.amazonaws.com/groups/s3/LogDelivery --grant-read-acp URI=http://
acs.amazonaws.com/groups/s3/LogDelivery

  # List the buckets in this account.
  buckets="$(aws s3 ls | awk '{print $3}')"
```

```
  # Put a bucket logging configuration on each bucket.
  for bucket in $buckets
      do
        # This if statement excludes the logging bucket.
        if [ "$bucket" == "$loggingBucket" ] ; then
            continue;
        fi
        printf '{
          "LoggingEnabled": {
            "TargetBucket": "%s",
            "TargetPrefix": "%s/"
        }
      }' "$loggingBucket" "$bucket"  > logging.json
      aws s3api put-bucket-logging --bucket $bucket --bucket-logging-status file://
logging.json
      echo "$bucket done"
  done

  rm logging.json

  echo "Complete"
```

Verifying your server access logs setup

After you enable server access logging, complete the following steps:

- Access the destination bucket and verify that the log files are being delivered. After the access logs are set up, Amazon S3 immediately starts capturing requests and logging them. However, it might take a few hours before the logs are delivered to the destination bucket. For more information, see the section called “Bucket logging status changes take effect over time” and the section called “Best-effort server log delivery”.

  You can also automatically verify log delivery by using Amazon S3 request metrics and setting up Amazon CloudWatch alarms for these metrics. For more information, see Monitoring metrics with Amazon CloudWatch.
- Verify that you are able to open and read the contents of the log files.

For server access logging troubleshooting information, see Troubleshoot server access logging.

Amazon S3 server access log format

Server access logging provides detailed records for the requests that are made to an Amazon S3 bucket. You can use server access logs for the following purposes:

- Performing security and access audits
- Learning about your customer base
- Understanding your Amazon S3 bill

This section describes the format and other details about Amazon S3 server access log files.

> **Note**
>
> This page covers the space-delimited text format used for logs delivered to an Amazon S3 general purpose bucket. For the structured JSON format used in Amazon CloudWatch Logs, see Log format in CloudWatch Logs.

Server access log files consist of a sequence of newline-delimited log records. Each log record represents one request and consists of space-delimited fields.

The following is an example log consisting of five log records.

> **Note**
>
> Any field can be set to `-` to indicate that the data was unknown or unavailable, or that the field was not applicable to this request.

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 3E57427F3EXAMPLE
 REST.GET.VERSIONING - "GET /amzn-s3-demo-bucket1?versioning HTTP/1.1" 200 - 113 - 7 -
 "-" "S3Console/0.4" - s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/
XV/VLi31234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-demo-bucket1.s3.us-
west-1.amazonaws.com TLSV1.2 arn:aws:s3:us-west-1:123456789012:accesspoint/example-AP
 Yes us-east-1
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
```

```
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 891CE47D2EXAMPLE
 REST.GET.LOGGING_STATUS - "GET /amzn-s3-demo-bucket1?logging HTTP/1.1" 200 -
 242 - 11 - "-" "S3Console/0.4" - 9vKBE6vMhrNiWHZmb2L0mXOcqPGzQOI5XLnCtZNPxev+Hf
+7tpT6sxDwDty4LHBUOZJG96N1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-
demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - - us-east-1
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:00:38 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be A1206F460EXAMPLE
 REST.GET.BUCKETPOLICY - "GET /amzn-s3-demo-bucket1?policy HTTP/1.1" 404
 NoSuchBucketPolicy 297 - 38 - "-" "S3Console/0.4" - BNaBsXZQQDbssi6xMBdBU2sLt
+Yf5kZDmeBUP35sFoKa3sLLeMC78iwEIWxs99CRUrbS4n11234= SigV4 ECDHE-RSA-AES128-GCM-SHA256
 AuthHeader amzn-s3-demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - Yes us-east-1
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:01:00 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 7B4A0FABBEXAMPLE
 REST.GET.VERSIONING - "GET /amzn-s3-demo-bucket1?versioning HTTP/1.1" 200 -
 113 - 33 - "-" "S3Console/0.4" - Ke1bUcazaN1jWuUlPJaxF64cQVpUEhoZKEG/hmy/gijN/
I1DeWqDfFvnpybfEseEME/u7ME1234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader amzn-s3-
demo-bucket1.s3.us-west-1.amazonaws.com TLSV1.2 - - us-east-1
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 amzn-s3-demo-bucket1 [06/Feb/2019:00:01:57 +0000] 192.0.2.3
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
 DD6CC733AEXAMPLE REST.PUT.OBJECT s3-dg.pdf "PUT /amzn-s3-demo-bucket1/
s3-dg.pdf HTTP/1.1" 200 - - 4406583 41754 28 "-" "S3Console/0.4" -
 10S62Zv81kBW7BB6SX4XJ48o6kpcl6LPwEoizZQQxJd5qDSCTLX0TgS37kYUBKQW3+bPdrg1234= SigV4
 ECDHE-RSA-AES128-SHA AuthHeader amzn-s3-demo-bucket1.s3.us-west-1.amazonaws.com
 TLSV1.2 - Yes us-east-1
```

The following is an example log record for the **Compute checksum** operation:

```
7cd47ef2be amzn-s3-demo-bucket [06/Feb/2019:00:00:38 +0000] -
 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be e5042925-b524-4b3b-
a869-f3881e78ff3a S3.COMPUTE.OBJECT.CHECKSUM example-object - - - - 1048576 - -
 - - - bPf7qjG4XwYdPgDQTl72GW/uotRhdPz2UryEyAFLDSRmKrakUkJCYLtAw6fdANcrsUYc1M/
kIulXM1u5vZQT5g== - - - - - - - -
```

Topics

- Log record fields
- Additional logging for copy operations
- Custom access log information
- Programming considerations for extensible server access log format

Log record fields

The following list describes the log record fields.

Bucket Owner

The canonical user ID of the owner of the source bucket. The canonical user ID is another form of the AWS account ID. For more information about the canonical user ID, see AWS account identifiers in the *AWS General Reference*. For information about how to find the canonical user ID for your account, see Finding the canonical user ID for your AWS account.

Example entry

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

Bucket

The name of the bucket that the request was processed against. If the system receives a malformed request and cannot determine the bucket, the request will not appear in any server access log.

Example entry

```
amzn-s3-demo-bucket1
```

Time

The time at which the request was received; these dates and times are in Coordinated Universal Time (UTC). The format, using `strftime()` terminology, is as follows: `[%d/%b/%Y:%H:%M:%S %z]`

Example entry

```
[06/Feb/2019:00:00:38 +0000]
```

Remote IP

The apparent IP address of the requester. Intermediate proxies and firewalls might obscure the actual IP address of the machine that's making the request.

Example entry

```
192.0.2.3
```

Requester

The canonical user ID of the requester, or a `-` for unauthenticated requests. If the requester was an IAM user, this field returns the requester's IAM user name along with the AWS account that the IAM user belongs to. This identifier is the same one used for access control purposes.

Example entry

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

If the requester is using an assumed role, this field returns the assumed IAM role.

Example entry

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

Request ID

A string generated by Amazon S3 to uniquely identify each request. For **Compute checksum** job requests, the **Request ID** field displays the associated job ID. For more information, see Compute checksums.

Example entry

```
3E57427F33A59F07
```

Operation

The operation listed here is declared as `SOAP.`*`operation`*, `REST.`*`HTTP_method`*`.`*`resource_type`*, `WEBSITE.`*`HTTP_method`*`.`*`resource_type`*, or `BATCH.DELETE.OBJECT`, or `S3.action.resource_type` for S3 Lifecycle and logging. For Compute checksum job requests, the operation is listed as `S3.COMPUTE.OBJECT.CHECKSUM`.

Example entry

```
REST.PUT.OBJECT
```

```
S3.COMPUTE.OBJECT.CHECKSUM
```

Key

The key (object name) part of the request.

Example entry

```
/photos/2019/08/puppy.jpg
```

Request-URI

The `Request-URI` part of the HTTP request message. This field may include unescaped quotes from the user input.

Example Entry

```
"GET /amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg?x-foo=bar HTTP/1.1"
```

HTTP status

The numeric HTTP status code of the response.

Example entry

```
200
```

Error Code

The Amazon S3 Error responses , or `-` if no error occurred.

Example entry

```
NoSuchBucket
```

Bytes Sent

The number of response bytes sent, excluding HTTP protocol overhead, or `-` if zero.

Example entry

```
2662992
```

Object Size

The total size of the object in question.

Example entry

```
3462992
```

Total Time

The number of milliseconds that the request was in flight from the server's perspective. This value is measured from the time that your request is received to the time that the last byte of the response is sent. Measurements made from the client's perspective might be longer because of network latency.

Example entry

```
70
```

Turn-Around Time

The number of milliseconds that Amazon S3 spent processing your request. This value is measured from the time that the last byte of your request was received until the time that the first byte of the response was sent.

Example entry

```
10
```

Referer

The value of the HTTP `Referer` header, if present. HTTP user-agents (for example, browsers) typically set this header to the URL of the linking or embedding page when making a request. This field may include unescaped quotes from the user input.

Example entry

```
"http://www.example.com/webservices"
```

User-Agent

The value of the HTTP `User-Agent` header. This field may include unescaped quotes from the user input.

Example entry

```
"curl/7.15.1"
```

Version Id

The version ID in the request, or `-` if the operation doesn't take a `versionId` parameter.

Example entry

```
3HL4kqtJvjVBH40Nrjfkd
```

Host Id

The `x-amz-id-2` or Amazon S3 extended request ID.

Example entry

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

Signature Version

The signature version, `SigV2` or `SigV4`, that was used to authenticate the request or a `-` for unauthenticated requests.

Example entry

```
SigV2
```

Cipher Suite

The Transport Layer Security (TLS) cipher that was negotiated for an HTTPS request or a `-` for HTTP.

Example entry

```
ECDHE-RSA-AES128-GCM-SHA256
```

Authentication Type

The type of request authentication used: `AuthHeader` for authentication headers, `QueryString` for query string (presigned URL), or a `-` for unauthenticated requests.

Example entry

```
AuthHeader
```

Host Header

The endpoint used to connect to Amazon S3.

Example entry

```
s3.us-west-2.amazonaws.com
```

Some earlier Regions support legacy endpoints. You might see these endpoints in your server access logs or AWS CloudTrail logs. For more information, see Legacy endpoints. For a complete list of Amazon S3 Regions and endpoints, see Amazon S3 endpoints and quotas in the *Amazon Web Services General Reference*.

TLS version

The Transport Layer Security (TLS) version negotiated by the client. The value is one of following: `TLSv1.1`, `TLSv1.2`, `TLSv1.3`, or `-` if TLS wasn't used.

Example entry

```
TLSv1.2
```

Access Point ARN

The Amazon Resource Name (ARN) of the access point of the request. If the access point ARN is malformed or not used, the field will contain a `-`. For more information about access points, see Using Amazon S3 access points for general purpose buckets. For more information about ARNs, see Amazon Resource Name (ARN) in the *AWS Reference Guide*.

Example entry

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

aclRequired

A string that indicates whether the request required an access control list (ACL) for authorization. If the request required an ACL for authorization, the string is `Yes`. If no ACLs were required, the string is `-`. For more information about ACLs, see Access control list (ACL) overview. For more information about using the `aclRequired` field to disable ACLs, see Controlling ownership of objects and disabling ACLs for your bucket.

Example entry

```
Yes
```

Source region

The AWS Region from which the request originated. This field shows a dash (-) when the origin region cannot be determined (such as PrivateLink connections, Direct Connect connections, Bring your own IP addresses (BYOIP), or non-AWS IP addresses) or when the log is generated by operations triggered based on customer-set policies or actions, such as lifecycle and checksum.

Example entry

```
us-east-1
```

Additional logging for copy operations

A copy operation involves a `GET` and a `PUT`. For that reason, we log two records when performing a copy operation. The previous section describes the fields related to the `PUT` part of the operation. The following list describes the fields in the record that relate to the `GET` part of the copy operation.

Bucket Owner

The canonical user ID of the bucket that stores the object being copied. The canonical user ID is another form of the AWS account ID. For more information about the canonical user ID, see AWS account identifiers in the *AWS General Reference*. For information about how to find the canonical user ID for your account, see Finding the canonical user ID for your AWS account.

Example entry

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

Bucket

The name of the bucket that stores the object that's being copied.

Example entry

```
amzn-s3-demo-bucket1
```

Time

The time at which the request was received; these dates and times are in Coordinated Universal Time (UTC). The format, using `strftime()` terminology, is as follows: `[%d/%B/%Y:%H:%M:%S %z]`

Example entry

```
[06/Feb/2019:00:00:38 +0000]
```

Remote IP

The apparent IP address of the requester. Intermediate proxies and firewalls might obscure the actual IP address of the machine that's making the request.

Example entry

```
192.0.2.3
```

Requester

The canonical user ID of the requester, or a `-` for unauthenticated requests. If the requester was an IAM user, this field will return the requester's IAM user name along with the AWS account root user that the IAM user belongs to. This identifier is the same one used for access control purposes.

Example entry

```
79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be
```

If the requester is using an assumed role, this field returns the assumed IAM role.

Example entry

```
arn:aws:sts::123456789012:assumed-role/roleName/test-role
```

Request ID

A string generated by Amazon S3 to uniquely identify each request. For **Compute checksum** job requests, the **Request ID** field displays the associated job ID. For more information, see Compute checksums.

Example entry

```
3E57427F33A59F07
```

Operation

The operation listed here is declared as `SOAP.operation`, `REST.HTTP_method.resource_type`, `WEBSITE.HTTP_method.resource_type`, or `BATCH.DELETE.OBJECT`.

Example entry

```
REST.COPY.OBJECT_GET
```

Key

The key (object name) of the object being copied, or `-` if the operation doesn't take a key parameter.

Example entry

```
/photos/2019/08/puppy.jpg
```

Request-URI

The `Request-URI` part of the HTTP request message. This field may include unescaped quotes from the user input.

Example entry

```
"GET /amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg?x-foo=bar"
```

HTTP status

The numeric HTTP status code of the GET portion of the copy operation.

Example entry

```
200
```

Error Code

The Amazon S3 Error responses of the GET portion of the copy operation, or - if no error occurred.

Example entry

```
NoSuchBucket
```

Bytes Sent

The number of response bytes sent, excluding the HTTP protocol overhead, or - if zero.

Example entry

```
2662992
```

Object Size

The total size of the object in question.

Example entry

```
3462992
```

Total Time

The number of milliseconds that the request was in flight from the server's perspective. This value is measured from the time that your request is received to the time that the last byte of the response is sent. Measurements made from the client's perspective might be longer because of network latency.

Example entry

```
70
```

Turn-Around Time

The number of milliseconds that Amazon S3 spent processing your request. This value is measured from the time that the last byte of your request was received until the time that the first byte of the response was sent.

Example entry

```
10
```

Referer

The value of the HTTP `Referer` header, if present. HTTP user-agents (for example, browsers) typically set this header to the URL of the linking or embedding page when making a request. This field may include unescaped quotes from the user input.

Example entry

```
"http://www.example.com/webservices"
```

User-Agent

The value of the HTTP `User-Agent` header. This field may include unescaped quotes from the user input.

Example entry

```
"curl/7.15.1"
```

Version Id

The version ID of the object being copied, or `-` if the `x-amz-copy-source` header didn't specify a `versionId` parameter as part of the copy source.

Example Entry

```
3HL4kqtJvjVBH40Nrjfkd
```

Host Id

The `x-amz-id-2` or Amazon S3 extended request ID.

Example entry

```
s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234=
```

Signature Version

The signature version, `SigV2` or `SigV4`, that was used to authenticate the request, or a `-` for unauthenticated requests.

Example entry

```
SigV4
```

Cipher Suite

The Transport Layer Security (TLS) cipher that was negotiated for an HTTPS request, or a `-` for HTTP.

Example entry

```
ECDHE-RSA-AES128-GCM-SHA256
```

Authentication Type

The type of request authentication used: `AuthHeader` for authentication headers, `QueryString` for query strings (presigned URLs), or a `-` for unauthenticated requests.

Example entry

```
AuthHeader
```

Host Header

The endpoint that was used to connect to Amazon S3.

Example entry

```
s3.us-west-2.amazonaws.com
```

Some earlier Regions support legacy endpoints. You might see these endpoints in your server access logs or AWS CloudTrail logs. For more information, see Legacy endpoints. For a complete list of Amazon S3 Regions and endpoints, see Amazon S3 endpoints and quotas in the *Amazon Web Services General Reference*.

TLS version

The Transport Layer Security (TLS) version negotiated by the client. The value is one of following: `TLSv1.1`, `TLSv1.2`, `TLSv1.3`, or `-` if TLS wasn't used.

Example entry

```
TLSv1.2
```

Access Point ARN

The Amazon Resource Name (ARN) of the access point of the request. If the access point ARN is malformed or not used, the field will contain a `-`. For more information about access points, see Using Amazon S3 access points for general purpose buckets. For more information about ARNs, see Amazon Resource Name (ARN) in the *AWS Reference Guide*.

Example entry

```
arn:aws:s3:us-east-1:123456789012:accesspoint/example-AP
```

aclRequired

A string that indicates whether the request required an access control list (ACL) for authorization. If the request required an ACL for authorization, the string is `Yes`. If no ACLs were required, the string is `-`. For more information about ACLs, see Access control list (ACL) overview. For more information about using the `aclRequired` field to disable ACLs, see Controlling ownership of objects and disabling ACLs for your bucket.

Example entry

```
Yes
```

Source region

The AWS Region from which the request originated. This field shows a dash (-) when the origin region cannot be determined (such as PrivateLink connections, Direct Connect connections,

Bring your own IP addresses (BYOIP), or non-AWS IP addresses) or when the log is generated by operations triggered based on customer-set policies or actions, such as lifecycle and checksum.

Example entry

```
us-east-1
```

Custom access log information

You can include custom information to be stored in the access log record for a request. To do this, add a custom query-string parameter to the URL for the request. Amazon S3 ignores query-string parameters that begin with `x-`, but includes those parameters in the access log record for the request, as part of the `Request-URI` field of the log record.

For example, a `GET` request for `"s3.amazonaws.com/amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg?x-user=johndoe"` works the same as the request for `"s3.amazonaws.com/amzn-s3-demo-bucket1/photos/2019/08/puppy.jpg"`, except that the `"x-user=johndoe"` string is included in the `Request-URI` field for the associated log record. This functionality is available in the REST interface only.

Programming considerations for extensible server access log format

Occasionally, we might extend the access log record format by adding new fields to the end of each line. Therefore, make sure that any of your code that parses server access logs can handle trailing fields that it might not understand.

Managing log retention

An Amazon S3 bucket with server access logging enabled can accumulate many server log objects over time. You can use Amazon S3 Lifecycle configuration to set rules so that Amazon S3 automatically deletes log objects after a specified period. If you specified a prefix in your logging configuration, you can scope the lifecycle rule to that prefix. For example, if your log objects have the prefix `logs/`, you can create a lifecycle rule to delete all objects with that prefix after a specified number of days. For more information, see Managing the lifecycle of objects.

For logs delivered to CloudWatch Logs, see Managing log retention.

Using Amazon S3 server access logs to identify requests

You can identify Amazon S3 requests by using Amazon S3 server access logs.

Note

- To identify Amazon S3 requests, we recommend that you use AWS CloudTrail data events instead of Amazon S3 server access logs. CloudTrail data events are easier to set up and contain more information. For more information, see Identifying Amazon S3 requests using CloudTrail.
- Depending on how many access requests you get, analyzing your logs might require more resources or time than using CloudTrail data events.

Topics

- Querying access logs for requests by using Amazon Athena
- Identifying Signature Version 2 requests
- Identifying object access requests

Querying access logs for requests by using Amazon Athena

You can identify Amazon S3 requests with Amazon S3 access logs by using Amazon Athena.

Amazon S3 stores server access logs as objects in an S3 bucket. It is often easier to use a tool that can analyze the logs in Amazon S3. Athena supports analysis of S3 objects and can be used to query Amazon S3 access logs.

Example

The following example shows how you can query Amazon S3 server access logs in Amazon Athena. Replace the *`user input placeholders`* used in the following examples with your own information.

Note

To specify an Amazon S3 location in an Athena query, you must provide an S3 URI for the bucket where your logs are delivered to. This URI must include the bucket name and prefix in the following format: `s3://`*`amzn-s3-demo-bucket1-logs/prefix`*`/`

1. Open the Athena console at https://console.aws.amazon.com/athena/.

2. In the Query Editor, run a command similar to the following. Replace *s3_access_logs_db* with the name that you want to give to your database.

```
CREATE DATABASE s3_access_logs_db
```

> **Note**
>
> It's a best practice to create the database in the same AWS Region as your S3 bucket.

3. In the Query Editor, run a command similar to the following to create a table schema in the database that you created in step 2. Replace *s3_access_logs_db.mybucket_logs* with the name that you want to give to your table. The `STRING` and `BIGINT` data type values are the access log properties. You can query these properties in Athena. For `LOCATION`, enter the S3 bucket and prefix path as noted earlier.

   Date-based partitioning

```
CREATE EXTERNAL TABLE s3_access_logs_db.mybucket_logs(
 `bucketowner` STRING,
 `bucket_name` STRING,
 `requestdatetime` STRING,
 `remoteip` STRING,
 `requester` STRING,
 `requestid` STRING,
 `operation` STRING,
 `key` STRING,
 `request_uri` STRING,
 `httpstatus` STRING,
 `errorcode` STRING,
 `bytessent` BIGINT,
 `objectsize` BIGINT,
 `totaltime` STRING,
 `turnaroundtime` STRING,
 `referrer` STRING,
 `useragent` STRING,
 `versionid` STRING,
 `hostid` STRING,
 `sigv` STRING,
 `ciphersuite` STRING,
 `authtype` STRING,
 `endpoint` STRING,
```

```
 `tlsversion` STRING,
 `accesspointarn` STRING,
 `aclrequired` STRING,
 `sourceregion` STRING)
 PARTITIONED BY (
   `timestamp` string)
ROW FORMAT SERDE
 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
 'input.regex'='([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) (-|[0-9]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) ([^ ]*)(?: ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*))?.*$')
STORED AS INPUTFORMAT
 'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
 'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
 's3://bucket-name/prefix-name/account-id/region/source-bucket-name/'
 TBLPROPERTIES (
  'projection.enabled'='true',
  'projection.timestamp.format'='yyyy/MM/dd',
  'projection.timestamp.interval'='1',
  'projection.timestamp.interval.unit'='DAYS',
  'projection.timestamp.range'='2024/01/01,NOW',
  'projection.timestamp.type'='date',
  'storage.location.template'='s3://bucket-name/prefix-name/account-id/region/
source-bucket-name/${timestamp}')
```

Non-date-based partitioning

```
CREATE EXTERNAL TABLE `s3_access_logs_db.mybucket_logs`(
  `bucketowner` STRING,
  `bucket_name` STRING,
  `requestdatetime` STRING,
  `remoteip` STRING,
  `requester` STRING,
  `requestid` STRING,
  `operation` STRING,
  `key` STRING,
  `request_uri` STRING,
  `httpstatus` STRING,
  `errorcode` STRING,
```

```
  `bytessent` BIGINT,
  `objectsize` BIGINT,
  `totaltime` STRING,
  `turnaroundtime` STRING,
  `referrer` STRING,
  `useragent` STRING,
  `versionid` STRING,
  `hostid` STRING,
  `sigv` STRING,
  `ciphersuite` STRING,
  `authtype` STRING,
  `endpoint` STRING,
  `tlsversion` STRING,
  `accesspointarn` STRING,
  `aclrequired` STRING,
  `sourceregion` STRING)
ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
  'input.regex'='([^ ]*) ([^ ]*) \\[(.*?)\\] ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) (-|[0-9]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) (\"[^\"]*\"|-) ([^ ]*)(?: ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*)
 ([^ ]*) ([^ ]*) ([^ ]*) ([^ ]*))?.*$')
STORED AS INPUTFORMAT
  'org.apache.hadoop.mapred.TextInputFormat'
OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
LOCATION
  's3://amzn-s3-demo-bucket1-logs/prefix/'
```

4. In the navigation pane, under **Database**, choose your database.
5. Under **Tables**, choose **Preview table** next to your table name.

   In the **Results** pane, you should see data from the server access logs, such as `bucketowner`, `bucket`, `requestdatetime`, and so on. This means that you successfully created the Athena table. You can now query the Amazon S3 server access logs.

Example— Show who deleted an object and when (timestamp, IP address, and IAM user)

```
SELECT requestdatetime, remoteip, requester, key
FROM s3_access_logs_db.mybucket_logs
WHERE key = 'images/picture.jpg' AND operation like '%DELETE%';
```

Example— Show all operations that were performed by an IAM user

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE requester='arn:aws:iam::123456789123:user/user_name';
```

Example— Show all operations that were performed on an object in a specific time period

```
SELECT *
FROM s3_access_logs_db.mybucket_logs
WHERE Key='prefix/images/picture.jpg'
AND parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2017-02-18:07:00:00','yyyy-MM-dd:HH:mm:ss')
AND parse_datetime('2017-02-18:08:00:00','yyyy-MM-dd:HH:mm:ss');
```

Example— Show how much data was transferred to a specific IP address in a specific time period

```
SELECT coalesce(SUM(bytessent), 0) AS bytessenttotal
FROM s3_access_logs_db.mybucket_logs
WHERE remoteip='192.0.2.1'
AND parse_datetime(requestdatetime,'dd/MMM/yyyy:HH:mm:ss Z')
BETWEEN parse_datetime('2022-06-01','yyyy-MM-dd')
AND parse_datetime('2022-07-01','yyyy-MM-dd');
```

Example— Find request IDs for HTTP 5xx errors in a specific time period

```
SELECT requestdatetime, key, httpstatus, errorcode, requestid, hostid
FROM s3_access_logs_db.mybucket_logs
WHERE httpstatus like '5%' AND timestamp
BETWEEN '2024/01/29'
AND '2024/01/30'
```

Identifying Signature Version 2 requests

You can use server access logs to identify Signature Version 2 requests. For details on querying logs with Amazon Athena, see the Athena querying section under Delivering logs to an Amazon S3 general purpose bucket.

Identifying object access requests

You can use server access logs to identify object access patterns. For details on querying logs with Amazon Athena, see the Athena querying section under Delivering logs to an Amazon S3 general purpose bucket.

Troubleshoot server access logging

The following topics can help you troubleshoot issues that you might encounter when setting up logging with Amazon S3.

Topics

- Common error messages when setting up logging
- Troubleshooting delivery failures

Common error messages when setting up logging

The following common error messages can appear when you're enabling logging through the AWS Command Line Interface (AWS CLI) and AWS SDKs:

Error: Cross S3 location logging not allowed

If the destination bucket (also known as a *target bucket*) is in a different Region than the source bucket, a Cross S3 location logging not allowed error occurs. To resolve this error, make sure that the destination bucket configured to receive the access logs is in the same AWS Region and AWS account as the source bucket.

Error: The owner for the bucket to be logged and the target bucket must be the same

When you're enabling server access logging, this error occurs if the specified destination bucket belongs to a different account. To resolve this error, make sure that the destination bucket is in the same AWS account as the source bucket.

> **Note**
>
> We recommend that you choose a destination bucket that's different from the source bucket. When the source bucket and destination bucket are the same, additional logs are created for the logs that are written to the bucket, which can increase your storage bill. These extra logs about logs can also make it difficult to find the particular logs that you're

looking for. For simpler log management, we recommend saving access logs in a different bucket.

Error: The target bucket for logging does not exist

The destination bucket must exist prior to setting the configuration. This error indicates that the destination bucket doesn't exist or can't be found. Make sure that the bucket name is spelled correctly, and then try again.

Error: Target grants not allowed for bucket owner enforced buckets

This error indicates that the destination bucket uses the Bucket owner enforced setting for S3 Object Ownership. The Bucket owner enforced setting doesn't support destination (target) grants. For more information, see Permissions for log delivery.

Troubleshooting delivery failures

To avoid server access logging issues, make sure that you're following these best practices:

- **The S3 log delivery group has write access to the destination bucket** – The S3 log delivery group delivers server access logs to the destination bucket. A bucket policy or bucket access control list (ACL) can be used to grant write access to the destination bucket. However, we recommend that you use a bucket policy instead of an ACL. For more information about how to grant write access to your destination bucket, see Permissions for log delivery.

  Note

  If the destination bucket uses the Bucket owner enforced setting for Object Ownership, be aware of the following:

  - ACLs are disabled and no longer affect permissions. This means that you can't update your bucket ACL to grant access to the S3 log delivery group. Instead, to grant access to the logging service principal, you must update the bucket policy for the destination bucket.
  - You can't include destination grants in your `PutBucketLogging` configuration.

- **The bucket policy for the destination bucket allows access to the logs** – Check the bucket policy of the destination bucket. Search the bucket policy for any statements that contain

`"Effect": "Deny"`. Then, verify that the `Deny` statement isn't preventing access logs from being written to the bucket.

- **S3 Object Lock isn't enabled on the destination bucket** – Check if the destination bucket has Object Lock enabled. Object Lock blocks server access log delivery. You must choose a destination bucket that doesn't have Object Lock enabled.
- **The destination bucket must use Amazon S3 managed keys (SSE-S3)** – If the destination bucket uses SSE-KMS default encryption, log objects might be created but encrypted with a key that you can't access.
- **The destination bucket does not have Requester Pays enabled** – Using a Requester Pays bucket as the destination bucket for server access logging is not supported. To allow delivery of server access logs, disable the Requester Pays option on the destination bucket.
- **Review your AWS Organizations service control policies (SCPs) and resource control policies (RCPs)** – When you're using AWS Organizations, check the service control policies and resource control policies to make sure that Amazon S3 access is allowed. These policies specify the maximum permissions for principals and resources in the affected accounts. Search the policies for any statements that contain `"Effect": "Deny"` and verify that `Deny` statements aren't preventing any access logs from being written to the bucket. For more information, see Authorization policies in AWS Organizations in the *AWS Organizations User Guide*.
- **Allow some time for recent logging configuration changes to take effect** – Enabling server access logging for the first time, or changing the destination bucket for logs, requires time to fully take effect. It might take longer than an hour for all requests to be properly logged and delivered.

  To check for log delivery failures, enable request metrics in Amazon CloudWatch. If the logs are not delivered within a few hours, look for the `4xxErrors` metric, which can indicate log delivery failures. For more information about enabling request metrics, see the section called “Creating a metrics configuration for all objects”.

Monitoring metrics with Amazon CloudWatch

Amazon CloudWatch metrics for Amazon S3 can help you understand and improve the performance of applications that use Amazon S3. There are several ways that you can use CloudWatch with Amazon S3.

Daily storage metrics for buckets

Monitor bucket storage using CloudWatch, which collects and processes storage data from Amazon S3 into readable, daily metrics. These storage metrics for Amazon S3 are reported once per day and are provided to all customers at no additional cost.

Request metrics

Monitor Amazon S3 requests to quickly identify and act on operational issues. The metrics are available at 1-minute intervals after some latency for processing. These CloudWatch metrics are billed at the same rate as the Amazon CloudWatch custom metrics. For information about CloudWatch pricing, see Amazon CloudWatch pricing. To learn how to opt in to getting these metrics, see CloudWatch metrics configurations.

When enabled, request metrics are reported for all object operations. By default, these 1-minute metrics are available at the Amazon S3 bucket level. You can also define a filter for the metrics using a shared prefix, object tag, or access point:

- **Access point** – Access points are named network endpoints that are attached to buckets and simplify managing data access at scale for shared datasets in S3. With the access point filter, you can gain insights into your access point usage. For more information about access points, see Monitoring and logging access points.
- **Prefix** – Although the Amazon S3 data model is a flat structure, you can use prefixes to infer a hierarchy. A prefix is similar to a directory name that enables you to group similar objects together in a bucket. The S3 console supports prefixes with the concept of folders. If you filter by prefix, objects that have the same prefix are included in the metrics configuration. For more information about prefixes, see Organizing objects using prefixes.
- **Tags** – Tags are key-value name pairs that you can add to objects. Tags help you find and organize objects easily. You can also use tags as a filter for metrics configurations so that only objects with those tags are included in the metrics configuration. For more information about object tags, see Tagging your objects.

To align these metrics to specific business applications, workflows, or internal organizations, you can filter on a shared prefix, object tag, or access point.

Replication metrics

Monitor the total number of S3 API operations that are pending replication, the total size of objects pending replication, the maximum replication time to the destination AWS Region, and the total number of operations that failed replication. Replication rules that have S3 Replication Time Control (S3 RTC) or S3 Replication metrics enabled will publish replication metrics.

For more information, see Monitoring replication with metrics, event notifications, and statuses or Meeting compliance requirements with S3 Replication Time Control.

Amazon S3 Storage Lens metrics

You can publish S3 Storage Lens usage and activity metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. S3 Storage Lens metrics are available in the `AWS/S3/Storage-Lens` namespace. The CloudWatch publishing option is available for S3 Storage Lens dashboards upgraded to *advanced metrics and recommendations*. You can enable the CloudWatch publishing option for a new or existing dashboard configuration in S3 Storage Lens.

For more information, see Monitor S3 Storage Lens metrics in CloudWatch.

All CloudWatch statistics are retained for a period of 15 months so that you can access historical information and gain a better perspective on how your web application or service is performing. For more information about CloudWatch, see What is Amazon CloudWatch? in the *Amazon CloudWatch User Guide*. You may need some additional configurations to your CloudWatch alarms, depending on your use cases. For example, you can use metric math expression to create an alarm. For more information, see Use CloudWatch metrics, Use metric math, Using Amazon CloudWatch alarms, and Create a CloudWatch alarm based on a metric math expression in the *Amazon CloudWatch User Guide*.

Best-effort CloudWatch metrics delivery

CloudWatch metrics are delivered on a best-effort basis. Most requests for an Amazon S3 object that have request metrics result in a data point being sent to CloudWatch.

The completeness and timeliness of metrics are not guaranteed. The data point for a particular request might be returned with a timestamp that is later than when the request was actually processed. The data point for a minute might be delayed before being available through CloudWatch, or it might not be delivered at all. CloudWatch request metrics give you an idea of the nature of traffic against your bucket in near-real time. It is not meant to be a complete accounting of all requests.

It follows from the best-effort nature of this feature that the reports available at the Billing & Cost Management Dashboard might include one or more access requests that do not appear in the bucket metrics.

For more information, see the following topics.

Topics

- Metrics and dimensions
- Accessing CloudWatch metrics
- CloudWatch metrics configurations

Metrics and dimensions

The storage metrics and dimensions that Amazon S3 sends to Amazon CloudWatch are listed in the following tables.

Best-effort CloudWatch metrics delivery

CloudWatch metrics are delivered on a best-effort basis. Most requests for an Amazon S3 object that have request metrics result in a data point being sent to CloudWatch.

The completeness and timeliness of metrics are not guaranteed. The data point for a particular request might be returned with a timestamp that is later than when the request was actually processed. The data point for a minute might be delayed before being available through CloudWatch, or it might not be delivered at all. CloudWatch request metrics give you an idea of the nature of traffic against your bucket in near-real time. It is not meant to be a complete accounting of all requests.

It follows from the best-effort nature of this feature that the reports available at the Billing & Cost Management Dashboard might include one or more access requests that do not appear in the bucket metrics.

Topics

- Amazon S3 daily storage metrics for buckets in CloudWatch
- Amazon S3 request metrics in CloudWatch
- S3 Replication metrics in CloudWatch
- S3 Storage Lens metrics in CloudWatch
- S3 Object Lambda request metrics in CloudWatch
- Amazon S3 dimensions in CloudWatch
- S3 Replication dimensions in CloudWatch
- S3 Storage Lens dimensions in CloudWatch
- S3 Object Lambda request dimensions in CloudWatch

- Amazon S3 usage metrics

Amazon S3 daily storage metrics for buckets in CloudWatch

The AWS/S3 namespace includes the following daily storage metrics for buckets.

Metric	Description
BucketSizeBytes	The amount of data in bytes that is stored in a bucket in the following storage classes: • Reduced Redundancy Storage (RRS) (REDUCED_REDUNDANCY) • S3 Express One Zone (EXPRESS_ONEZONE) • S3 Glacier Deep Archive (DEEP_ARCHIVE) • S3 Glacier Flexible Retrieval (GLACIER) • S3 Glacier Instant Retrieval (GLACIER_IR) • S3 Intelligent-Tiering (INTELLIGENT_TIERING) • S3 One Zone-Infrequent Access (ONEZONE_IA) • S3 Standard (STANDARD) • S3 Standard-Infrequent Access (STANDARD_IA) This value is calculated by summing the size of all objects and metadata (such as bucket names) in the bucket (both current and noncurrent objects), including the size of all parts for all incomplete multipart uploads to the bucket. **Note** The S3 Express One Zone storage class is available only for directory buckets. Valid storage-type filters (see the StorageType dimension): • Reduced Redundancy Storage (RRS): ReducedRedundancyStorage

Metric	Description
	• S3 Express One Zone: `ExpressOneZoneStorage` • S3 Glacier Deep Archive: `DeepArchiveObjectOverhead` , `DeepArchiveS3ObjectOverhead` , `DeepArchiveStagingStorage` , `DeepArchiveStorage` • S3 Glacier Flexible Retrieval: `GlacierObjectOverhead` , `GlacierS3ObjectOverhead` , `GlacierStagingStorage` , `GlacierStorage` • S3 Glacier Instant Retrieval: `GlacierInstantRetrievalStorage` , `GlacierIRSizeOverhead` • S3 Intelligent-Tiering: `IntelligentTieringAAStorage` , `IntelligentTieringAIAStorage` , `IntelligentTieringDAAStorage` , `IntelligentTieringFAStorage` , `IntelligentTieringIAStorage` • S3 One Zone-Infrequent Access: `OneZoneIASizeOverhead` , `OneZoneIAStorage` • S3 Standard: `StandardStorage` • S3 Standard-Infrequent Access: `StandardIAObjectOverhead` , `StandardIASizeOverhead` , `StandardIAStorage` Units: Bytes Valid statistics: Average For more information about the `StorageType` dimensions, see the section called “Amazon S3 dimensions in CloudWatch”.

Metric	Description
NumberOfObjects	The total number of objects stored in a general purpose bucket for all storage classes. This value is calculated by counting all objects in the bucket, which includes current and noncurrent objects, delete markers, and the total number of parts for all incomplete multipart uploads to the bucket. For directory buckets with objects in the S3 Express One Zone storage class, this value is calculated by counting all objects in the bucket, but it doesn't include incomplete multipart uploads to the bucket. Valid storage type filters: AllStorageTypes (see the StorageType dimension) Units: Count Valid statistics: Average

Amazon S3 request metrics in CloudWatch

The AWS/S3 namespace includes the following request metrics. These metrics include non-billable requests (in the case of GET requests from CopyObject and Replication).

Metric	Description
AllRequests	The total number of HTTP requests made to an Amazon S3 bucket, regardless of type. If you're using a metrics configuration with a filter, then this metric returns only the HTTP requests that meet the filter's requirements. Units: Count Valid statistics: Sum
GetRequests	The number of HTTP GET requests made for objects in an Amazon S3 bucket. This doesn't include list operations. This metric is incremented for the source of each CopyObject request.

Metric	Description
	Units: Count Valid statistics: Sum **Note** Paginated list-oriented requests, such as ListMultipartUploads, ListParts, ListObjectVersions, and others, are not included in this metric.
`PutRequests`	The number of HTTP `PUT` requests made for objects in an Amazon S3 bucket. This metric is incremented for the destination of each `CopyObject` request. Units: Count Valid statistics: Sum
`DeleteRequests`	The number of HTTP `DELETE` requests made for objects in an Amazon S3 bucket. This metric also includes DeleteObjects requests. This metric shows the number of requests made, not the number of objects deleted. Units: Count Valid statistics: Sum
`HeadRequests`	The number of HTTP `HEAD` requests made to an Amazon S3 bucket. Units: Count Valid statistics: Sum

Metric	Description
PostRequests	The number of HTTP POST requests made to an Amazon S3 bucket. Units: Count Valid statistics: Sum **Note** DeleteObjects and SelectObjectContent requests are not included in this metric.
SelectRequests	The number of Amazon S3 SelectObjectContent requests made for objects in an Amazon S3 bucket. Units: Count Valid statistics: Sum
SelectBytesScanned	The number of bytes of data scanned with Amazon S3 SelectObjectContent requests in an Amazon S3 bucket. Units: Bytes Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9
SelectBytesReturned	The number of bytes of data returned with Amazon S3 SelectObjectContent requests in an Amazon S3 bucket. Units: Bytes Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9

Metric	Description
`ListRequests`	The number of HTTP requests that list the contents of a bucket. Units: Count Valid statistics: Sum
`BytesDownloaded`	The number of bytes downloaded for requests made to an Amazon S3 bucket, where the response includes a body. Units: Bytes Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9
`BytesUploaded`	The number of bytes uploaded for requests made to an Amazon S3 bucket, where the request includes a body. Units: Bytes Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9
`4xxErrors`	The number of HTTP 4*xx* client error status code requests made to an Amazon S3 bucket with a value of either 0 or 1. The Average statistic shows the error rate, and the Sum statistic shows the count of that type of error, during each period. Units: Count Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count

Metric	Description
`5xxErrors`	The number of HTTP 5*xx* server error status code requests made to an Amazon S3 bucket with a value of either 0 or 1. The Average statistic shows the error rate, and the Sum statistic shows the count of that type of error, during each period. Units: Count Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count
`FirstByteLatency`	The per-request time from the complete request being received by an Amazon S3 bucket to when the response starts to be returned. Units: Milliseconds Valid statistics: Average, Sum, Min, Max (same as p100), Sample Count, any percentile between p0.0 and p100
`TotalRequestLatency`	The elapsed per-request time from the first byte received to the last byte sent to an Amazon S3 bucket. This metric includes the time taken to receive the request body and send the response body, which is not included in `FirstByteLatency` . Units: Milliseconds Valid statistics: Average, Sum, Min, Max (same as p100), Sample Count, any percentile between p0.0 and p100

S3 Replication metrics in CloudWatch

You can monitor the progress of replication with S3 Replication metrics by tracking bytes pending, operations pending, and replication latency. For more information, see Monitoring progress with replication metrics.

Note

You can enable alarms for your replication metrics in Amazon CloudWatch. When you set up alarms for your replication metrics, set the **Missing data treatment** field to **Treat missing data as ignore (maintain the alarm state)**.

Metric	Description
`ReplicationLatency`	The maximum number of seconds by which the replication destination AWS Region is behind the source AWS Region for a given replication rule. Units: Seconds Valid statistics: Max
`BytesPendingReplication`	The total number of bytes of objects pending replication for a given replication rule. Units: Bytes Valid statistics: Max
`OperationsPendingReplication`	The number of operations pending replication for a given replication rule. Units: Count Valid statistics: Max
`OperationsFailedReplication`	The number of operations that failed to replicate for a given replication rule. Units: Count Valid statistics: Sum (total number of failed operations), Average (failure rate), Sample Count (total number of replication operations)

S3 Storage Lens metrics in CloudWatch

You can publish S3 Storage Lens usage and activity metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. S3 Storage Lens metrics are published to the `AWS/S3/Storage-Lens` namespace in CloudWatch. The CloudWatch publishing option is available for S3 Storage Lens dashboards that have been upgraded to advanced metrics and recommendations.

For a list of S3 Storage Lens metrics that are published to CloudWatch, see Amazon S3 Storage Lens metrics glossary. For a complete list of dimensions, see Dimensions.

S3 Object Lambda request metrics in CloudWatch

S3 Object Lambda includes the following request metrics.

Metric	Description
`AllRequests`	The total number of HTTP requests made to an Amazon S3 bucket by using an Object Lambda Access Point. Units: Count Valid statistics: Sum
`GetRequests`	The number of HTTP `GET` requests made for objects by using an Object Lambda Access Point. This metric does not include list operations. Units: Count Valid statistics: Sum
`BytesUploaded`	The number of bytes uploaded to an Amazon S3 bucket by using an Object Lambda Access Point, where the request includes a body. Units: Bytes Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9

Metric	Description
`PostRequests`	The number of HTTP `POST` requests made to an Amazon S3 bucket by using an Object Lambda Access Point. Units: Count Valid statistics: Sum
`PutRequests`	The number of HTTP `PUT` requests made for objects in an Amazon S3 bucket by using an Object Lambda Access Point. Units: Count Valid statistics: Sum
`DeleteRequests`	The number of HTTP `DELETE` requests made for objects in an Amazon S3 bucket by using an Object Lambda Access Point. This metric includes DeleteObjects requests. This metric shows the number of requests made, not the number of objects deleted. Units: Count Valid statistics: Sum
`BytesDownloaded`	The number of bytes downloaded for requests made to an Amazon S3 bucket by using an Object Lambda Access Point, where the response includes a body. Units: Bytes Valid statistics: Average (bytes per request), Sum (bytes per period), Sample Count, Min, Max (same as p100), any percentile between p0.0 and p99.9

Metric	Description
`FirstByteLatency`	The per-request time from the complete request being received by an Amazon S3 bucket through an Object Lambda Access Point to when the response starts to be returned. This metric is dependent on the AWS Lambda function's running time to transform the object before the function returns the bytes to the Object Lambda Access Point. Units: Milliseconds Valid statistics: Average, Sum, Min, Max (same as p100), Sample Count, any percentile between p0.0 and p100
`TotalRequestLatency`	The elapsed per-request time from the first byte received to the last byte sent to an Object Lambda Access Point. This metric includes the time taken to receive the request body and send the response body, which is not included in `FirstByteLatency` . Units: Milliseconds Valid statistics: Average, Sum, Min, Max (same as p100), Sample Count, any percentile between p0.0 and p100
`HeadRequests`	The number of HTTP `HEAD` requests made to an Amazon S3 bucket by using an Object Lambda Access Point. Units: Count Valid statistics: Sum
`ListRequests`	The number of HTTP `GET` requests that list the contents of an Amazon S3 bucket. This metric includes both `ListObjects` and `ListObjectsV2` operations. Units: Count Valid statistics: Sum

Metric	Description
4xxErrors	The number of HTTP 4*xx* client error status code requests made to an Amazon S3 bucket by using an Object Lambda Access Point with a value of either 0 or 1. The Average statistic shows the error rate, and the Sum statistic shows the count of that type of error, during each period. Units: Count Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count
5xxErrors	The number of HTTP 5*xx* server error status code requests made to an Amazon S3 bucket by using an Object Lambda Access Point with a value of either 0 or 1. The Average statistic shows the error rate, and the Sum statistic shows the count of that type of error, during each period. Units: Count Valid statistics: Average (reports per request), Sum (reports per period), Min, Max, Sample Count
ProxiedRequests	The number of HTTP requests to an Object Lambda Access Point that return the standard Amazon S3 API response. (Such requests do not have a Lambda function configured.) Units: Count Valid statistics: Sum
InvokedLambda	The number of HTTP requests to an S3 object where a Lambda function was invoked. Units: Count Valid statistics: Sum

Metric	Description
LambdaResponseRequests	The number of WriteGetObjectResponse requests made by the Lambda function. This metric applies only to GetObject requests.
LambdaResponse4xx	The number of HTTP 4*xx* client errors that occur when calling WriteGetObjectResponse from a Lambda function. This metric provides the same information as 4xxErrors, but only for WriteGetObjectResponse calls.
LambdaResponse5xx	The number of HTTP 5*xx* server errors that occur when calling WriteGetObjectResponse from a Lambda function. This metric provides the same information as 5xxErrors, but only for WriteGetObjectResponse calls.

Amazon S3 dimensions in CloudWatch

The following dimensions are used to filter Amazon S3 metrics.

Dimension	Description
BucketName	This dimension filters the data that you request for the identified bucket only.
StorageType	This dimension filters the data that you have stored in a bucket by the following types of storage: • DeepArchiveObjectOverhead – For each archived object, S3 Glacier adds 32 KB of storage for index and related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Glacier Deep Archive rates for this additional storage. • DeepArchiveS3ObjectOverhead – For each object archived to S3 Glacier Deep Archive, Amazon S3 uses 8 KB of storage for the name of the object and other metadata. You are charged S3 Standard rates for this additional storage.

Dimension	Description
	• `DeepArchiveStagingStorage` – The number of bytes used for parts of multipart upload objects before the `CompleteMultipartUpload` request is completed on objects in the S3 Glacier Deep Archive storage class. • `DeepArchiveStorage` – The number of bytes used for objects in the S3 Glacier Deep Archive storage class. • `ExpressOneZoneStorage` – The number of bytes used for objects in the S3 Express One Zone storage class. • `GlacierInstantRetrievalStorage` – The number of bytes used for objects in the S3 Glacier Instant Retrieval storage class. • `GlacierIRSizeOverhead` – For objects smaller than 128 KB stored in the S3 Glacier Instant Retrieval storage class, this storage type represents the additional bytes charged to meet the minimum object size requirement. • `GlacierObjectOverhead` – For each archived object, S3 Glacier adds 32 KB of storage for index and related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Glacier Flexible Retrieval rates for this additional storage. • `GlacierS3ObjectOverhead` – For each object archived to S3 Glacier Flexible Retrieval, Amazon S3 uses 8 KB of storage for the name of the object and other metadata. You are charged S3 Standard rates for this additional storage. • `GlacierStagingStorage` – The number of bytes used for parts of multipart upload objects before the `CompleteMultipartUpload` request is completed on objects in the S3 Glacier Flexible Retrieval storage class. • `GlacierStorage` – The number of bytes used for objects in the S3 Glacier Flexible Retrieval storage class. • `IntAAObjectOverhead` – For each object in the `INTELLIGENT_TIERING` storage class in the Archive Access tier, S3 Glacier adds 32 KB of storage for index and

Dimension	Description
	related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Glacier Flexible Retrieval rates for this additional storage. • `IntAAS3ObjectOverhead` – For each object in the `INTELLIGENT_TIERING` storage class in the Archive Access tier, Amazon S3 uses 8 KB of storage for the name of the object and other metadata. You are charged S3 Standard rates for this additional storage. • `IntDAAObjectOverhead` – For each object in the `INTELLIGENT_TIERING` storage class in the Deep Archive Access tier, S3 Glacier adds 32 KB of storage for index and related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Glacier Deep Archive storage rates for this additional storage. • `IntDAAS3ObjectOverhead` – For each object in the `INTELLIGENT_TIERING` storage class in the Deep Archive Access tier, Amazon S3 adds 8 KB of storage for index and related metadata. This extra data is necessary to identify and restore your object. You are charged S3 Standard rates for this additional storage. • `IntelligentTieringAAStorage` – The number of bytes used for objects in the Archive Access tier of the `INTELLIGENT_TIERING` storage class. • `IntelligentTieringAIAStorage` – The number of bytes used for objects in the Archive Instant Access tier of the `INTELLIGENT_TIERING` storage class. • `IntelligentTieringDAAStorage` – The number of bytes used for objects in the Deep Archive Access tier of the `INTELLIGENT_TIERING` storage class. • `IntelligentTieringFAStorage` – The number of bytes used for objects in the Frequent Access tier of the `INTELLIGENT_TIERING` storage class.

Dimension	Description
	• `IntelligentTieringIAStorage` – The number of bytes used for objects in the Infrequent Access tier of the `INTELLIGENT_TIERING` storage class. • `OneZoneIASizeOverhead` – For objects smaller than 128 KB stored in the S3 One Zone-Infrequent Access (`ONEZONE_IA`) storage class, this storage type represents the additional bytes charged to meet the minimum object size requirement. • `OneZoneIAStorage` – The number of bytes used for objects in the S3 One Zone-Infrequent Access (`ONEZONE_IA`) storage class. • `ReducedRedundancyStorage` – The number of bytes used for objects in the Reduced Redundancy Storage (RRS) class. • `StandardIASizeOverhead` – For objects smaller than 128 KB stored in the `STANDARD_IA` storage class, this storage type represents the additional bytes charged to meet the minimum object size requirement. • `StandardIAStorage` – The number of bytes used for objects in the S3 Standard-Infrequent Access (`STANDARD_IA`) storage class. • `StandardStorage` – The number of bytes used for objects in the `STANDARD` storage class.
`FilterId`	This dimension filters metrics configurations that you specify for the request metrics on a bucket. When you create a metrics configuration, you specify a filter ID (for example, a prefix, a tag, or an access point). For more information, see Creating a metrics configuration.

S3 Replication dimensions in CloudWatch

The following dimensions are used to filter S3 Replication metrics.

Dimension	Description
SourceBucket	The name of the bucket objects are replicated from.
DestinationBucket	The name of the bucket objects are replicated to.
RuleId	A unique identifier for the rule that triggered this replication metric to update.

S3 Storage Lens dimensions in CloudWatch

For a list of dimensions that are used to filter S3 Storage Lens metrics in CloudWatch, see Dimensions.

S3 Object Lambda request dimensions in CloudWatch

The following dimensions are used to filter data from an Object Lambda Access Point.

Dimension	Description
AccessPointName	The name of the access point of which requests are being made.
DataSourceARN	The source the Object Lambda Access Point is retrieving the data from. If the request invokes a Lambda function this refers to the Lambda Amazon Resource Name (ARN). Otherwise this refers to the access point ARN.

Amazon S3 usage metrics

You can use CloudWatch usage metrics to provide visibility into your account's usage of resources. Use these metrics to visualize your current service usage on CloudWatch graphs and dashboards.

Amazon S3 usage metrics correspond to AWS service quotas. You can configure alarms that alert you when your usage approaches a service quota. For more information about CloudWatch integration with service quotas, see AWS usage metrics in the *Amazon CloudWatch User Guide*.

Amazon S3 publishes the following metrics in the AWS/Usage namespace.

Metric	Description
`ResourceCount`	The number of the specified resources running in your account. The resources are defined by the dimensions associated with the metric.

The following dimensions are used to refine the usage metrics that are published by Amazon S3.

Dimension	Description
`Service`	The name of the AWS service containing the resource. For Amazon S3 usage metrics, the value for this dimension is S3.
`Type`	The type of entity that is being reported. Currently, the only valid value for Amazon S3 usage metrics is `Resource`.
`Resource`	The type of resource that is running. Currently, the only valid value for Amazon S3 usage metrics is `GeneralPurposeBuckets` , which returns the number of general purpose buckets in an AWS account. General purpose buckets allow objects that are stored across all storage classes, except S3 Express One Zone.

Accessing CloudWatch metrics

You can use the following procedures to view the storage metrics for Amazon S3. To get the Amazon S3 metrics involved, you must set a start and end timestamp. For metrics for any given 24-hour period, set the time period to 86400 seconds, the number of seconds in a day. Also, remember to set the `BucketName` and `StorageType` dimensions.

Using the AWS CLI

For example, if you want to use the AWS CLI to get the average of a specific bucket's size in bytes, you could use the following command:

```
aws cloudwatch get-metric-statistics --metric-name BucketSizeBytes --namespace AWS/S3
 --start-time 2016-10-19T00:00:00Z --end-time 2016-10-20T00:00:00Z --statistics Average
```

```
 --unit Bytes --region us-west-2 --dimensions Name=BucketName,Value=amzn-s3-demo-bucket
 Name=StorageType,Value=StandardStorage --period 86400 --output json
```

This example produces the following output.

```
{
    "Datapoints": [
        {
            "Timestamp": "2016-10-19T00:00:00Z",
            "Average": 1025328.0,
            "Unit": "Bytes"
        }
    ],
    "Label": "BucketSizeBytes"
}
```

Using the S3 console

To view metrics by using the Amazon CloudWatch console

1. Open the CloudWatch console at https://console.aws.amazon.com/cloudwatch/.
2. In the left navigation pane, choose **Metrics**.
3. Choose the **S3** namespace.
4. (Optional) To view a metric, enter the metric name in the search box.
5. (Optional) To filter by the **StorageType** dimension, enter the name of the storage class in the search box.

To view a list of valid metrics stored for your AWS account by using the AWS CLI

- At a command prompt, use the following command.

```
aws cloudwatch list-metrics --namespace "AWS/S3"
```

For more information about the permissions required to access CloudWatch dashboards, see Amazon CloudWatch dashboard permissions in the *Amazon CloudWatch User Guide*.

CloudWatch metrics configurations

With Amazon CloudWatch request metrics for Amazon S3, you can receive 1-minute CloudWatch metrics, set CloudWatch alarms, and access CloudWatch dashboards to view near-real-time operations and performance of your Amazon S3 storage. For applications that depend on cloud storage, these metrics let you quickly identify and act on operational issues. When enabled, these 1-minute metrics are available at the Amazon S3 bucket-level, by default.

If you want to get the CloudWatch request metrics for the objects in a bucket, you must create a metrics configuration for the bucket. For more information, see Creating a CloudWatch metrics configuration for all the objects in your bucket.

You can also use a shared prefix, object tags, or an access point to define a filter for the metrics collected. This method of defining a filter allows you to align metrics filters to specific business applications, workflows, or internal organizations. For more information, see Creating a metrics configuration that filters by prefix, object tag, or access point. For more information about the CloudWatch metrics that are available and the differences between storage and request metrics, see Monitoring metrics with Amazon CloudWatch.

Keep the following in mind when using metrics configurations:

- You can have a maximum of 1,000 metrics configurations per bucket.
- You can choose which objects in a bucket to include in metrics configurations by using filters. You can filter on a shared prefix, object tag, or access point to align metrics filters to specific business applications, workflows, or internal organizations. To request metrics for the entire bucket, create a metrics configuration without a filter.
- Metrics configurations are necessary only to enable request metrics. Bucket-level daily storage metrics are always turned on, and are provided at no additional cost. Currently, it's not possible to get daily storage metrics for a filtered subset of objects.
- Each metrics configuration enables the full set of available request metrics. Operation-specific metrics (such as `PostRequests`) are reported only if there are requests of that type for your bucket or your filter.
- Request metrics are reported for object-level operations. They are also reported for operations that list bucket contents, like GET Bucket (List Objects), GET Bucket Object Versions, and List Multipart Uploads, but they are not reported for other operations on buckets.
- Request metrics support filtering by prefix, object tags, or access point, but storage metrics do not.

Best-effort CloudWatch metrics delivery

CloudWatch metrics are delivered on a best-effort basis. Most requests for an Amazon S3 object that have request metrics result in a data point being sent to CloudWatch.

The completeness and timeliness of metrics are not guaranteed. The data point for a particular request might be returned with a timestamp that is later than when the request was actually processed. The data point for a minute might be delayed before being available through CloudWatch, or it might not be delivered at all. CloudWatch request metrics give you an idea of the nature of traffic against your bucket in near-real time. It is not meant to be a complete accounting of all requests.

It follows from the best-effort nature of this feature that the reports available at the Billing & Cost Management Dashboard might include one or more access requests that do not appear in the bucket metrics.

For more information about working with CloudWatch metrics in Amazon S3, see the following topics.

Topics

- Creating a CloudWatch metrics configuration for all the objects in your bucket
- Creating a metrics configuration that filters by prefix, object tag, or access point
- Deleting a metrics filter

Creating a CloudWatch metrics configuration for all the objects in your bucket

When you configure request metrics, you can create a CloudWatch metrics configuration for all the objects in your bucket, or you can filter by prefix, object tag, or access point. The procedures in this topic show you how to create a configuration for all the objects in your bucket. To create a configuration that filters by object tag, prefix, or access point, see Creating a metrics configuration that filters by prefix, object tag, or access point.

There are three types of Amazon CloudWatch metrics for Amazon S3: storage metrics, request metrics, and replication metrics. Storage metrics are reported once per day and are provided to all customers at no additional cost. Request metrics are available at one-minute intervals after some latency for processing. Request metrics are billed at the standard CloudWatch rate. You must opt in to request metrics by configuring them in the console or using the Amazon S3 API. S3 Replication metrics provide detailed metrics for the replication rules in your replication configuration. With

replication metrics, you can monitor minute-by-minute progress by tracking bytes pending, operations pending, operations that failed replication, and replication latency.

For more information about CloudWatch metrics for Amazon S3, see Monitoring metrics with Amazon CloudWatch.

You can add metrics configurations to a bucket using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), or the Amazon S3 REST API.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that contains the objects you want request metrics for.
4. Choose the **Metrics** tab.
5. Under **Bucket metrics**, choose **View additional charts**.
6. Choose the **Request metrics** tab.
7. Choose **Create filter**.
8. In the **Filter name** box, enter your filter name.

   Names can only contain letters, numbers, periods, dashes, and underscores. We recommend using the name `EntireBucket` for a filter that applies to all objects.
9. Under **Filter scope**, choose **This filter applies to all objects in the bucket**.

   You can also define a filter so that the metrics are only collected and reported on a subset of objects in the bucket. For more information, see Creating a metrics configuration that filters by prefix, object tag, or access point.
10. Choose **Save changes**.
11. On the **Request metrics** tab, under **Filters**, choose the filter that you just created.

    After about 15 minutes, CloudWatch begins tracking these request metrics. You can see them on the **Request metrics** tab. You can see graphs for the metrics on the Amazon S3 or CloudWatch console. Request metrics are billed at the standard CloudWatch rate. For more information, see Amazon CloudWatch pricing.

Using the REST API

You can also add metrics configurations programmatically with the Amazon S3 REST API. For more information about adding and working with metrics configurations, see the following topics in the *Amazon Simple Storage Service API Reference*:

- PUT Bucket Metric Configuration
- GET Bucket Metric Configuration
- List Bucket Metric Configuration
- DELETE Bucket Metric Configuration

Using the AWS CLI

1. Install and set up the AWS CLI. For instructions, see Installing, updating, and uninstalling the AWS CLI in the *AWS Command Line Interface User Guide*.
2. Open a terminal.
3. Run the following command to add a metrics configuration.

```
aws s3api put-bucket-metrics-configuration --endpoint https://s3.us-
west-2.amazonaws.com --bucket bucket-name --id metrics-config-id --metrics-
configuration '{"Id":"metrics-config-id"}'
```

Creating a metrics configuration that filters by prefix, object tag, or access point

There are three types of Amazon CloudWatch metrics for Amazon S3: storage metrics, request metrics, and replication metrics. Storage metrics are reported once per day and are provided to all customers at no additional cost. Request metrics are available at one-minute intervals after some latency for processing. Request metrics are billed at the standard CloudWatch rate. You must opt in to request metrics by configuring them in the console or using the Amazon S3 API. S3 Replication metrics provide detailed metrics for the replication rules in your replication configuration. With replication metrics, you can monitor minute-by-minute progress by tracking bytes pending, operations pending, operations that failed replication, and replication latency.

For more information about CloudWatch metrics for Amazon S3, see Monitoring metrics with Amazon CloudWatch.

When you configure CloudWatch metrics, you can create a filter for all the objects in your bucket, or you can filter the configuration into groups of related objects within a single bucket. You can filter objects in a bucket for inclusion in a metrics configuration based on one or more of the following filter types:

- **Object key name prefix** – Although the Amazon S3 data model is a flat structure, you can infer a hierarchy by using a prefix. The Amazon S3 console supports these prefixes with the concept of folders. If you filter by prefix, objects that have the same prefix are included in the metrics configuration. For more information about prefixes, see Organizing objects using prefixes.
- **Tag** – You can add tags, which are key-value name pairs, to objects. Tags help you find and organize objects easily. You can also use tags as filters for metrics configurations. For more information about object tags, see Tagging your objects.
- **Access point** – S3 Access Points are named network endpoints that are attached to buckets and simplify managing data access at scale for shared datasets in S3. When you create an access point filter, Amazon S3 includes requests to the access point that you specify in the metrics configuration. For more information, see Monitoring and logging access points.

> **Note**
>
> When you create a metrics configuration that filters by access point, you must use the access point Amazon Resource Name (ARN), not the access point alias. Make sure that you use the ARN for the access point itself, not the ARN for a specific object. For more information about access point ARNs, see Using Amazon S3 access points for general purpose buckets.

If you specify a filter, only requests that operate on single objects can match the filter and be included in the reported metrics. Requests like DeleteObjects and `ListObjects` requests don't return any metrics for configurations with filters.

To request more complex filtering, choose two or more elements. Only objects that have all of those elements are included in the metrics configuration. If you don't set filters, all of the objects in the bucket are included in the metrics configuration.

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **General purpose buckets**
3. In the buckets list, choose the name of the bucket that contains the objects that you want request metrics for.
4. Choose the **Metrics** tab.
5. Under **Bucket metrics**, choose **View additional charts**.
6. Choose the **Request metrics** tab.
7. Choose **Create filter**.
8. In the **Filter name** box, enter your filter name.

   Names can contain only letters, numbers, periods, dashes, and underscores.
9. Under **Filter scope**, choose **Limit the scope of this filter using a prefix, object tags, and an S3 Access Point, or a combination of all three**.
10. Under **Filter type**, choose at least one filter type: **Prefix**, **Object tags**, or **Access Point**.
11. To define a prefix filter and limit the scope of the filter to a single path, in the **Prefix** box, enter a prefix.
12. To define an object tags filter, under **Object tags**, choose **Add tag**, and then enter a tag **Key** and **Value**.
13. To define an access point filter, in the **S3 Access Point** field, enter the access point ARN, or choose **Browse S3** to navigate to the access point.

    > **Important**
    >
    > You cannot enter an access point alias. You must enter the ARN for the access point itself, not the ARN for a specific object.
14. Choose **Save changes**.

    Amazon S3 creates a filter that uses the prefix, tags, or access point that you specified.
15. On the **Request metrics** tab, under **Filters**, choose the filter that you just created.

    You have now created a filter that limits the request metrics scope by prefix, object tags, or access point. About 15 minutes after CloudWatch begins tracking these request metrics, you can see charts for the metrics on both the Amazon S3 and CloudWatch consoles. Request metrics are billed at the standard CloudWatch rate. For more information, see Amazon CloudWatch pricing.

You can also configure request metrics at the bucket level. For information, see Creating a CloudWatch metrics configuration for all the objects in your bucket.

Using the AWS CLI

1. Install and set up the AWS CLI. For instructions, see Installing, updating, and uninstalling the AWS CLI in the *AWS Command Line Interface User Guide*.
2. Open a terminal.
3. To add a metrics configuration, run one of the following commands:

Example: To filter by prefix

```
aws s3api put-bucket-metrics-configuration --bucket amzn-s3-demo-bucket --
id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
{"Prefix":"prefix1"}} '
```

Example: To filter by tags

```
aws s3api put-bucket-metrics-configuration --bucket amzn-s3-demo-bucket --
id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
{"Tag": {"Key": "string", "Value": "string"}} '
```

Example: To filter by access point

```
aws s3api put-bucket-metrics-configuration --bucket amzn-s3-demo-bucket --
id metrics-config-id --metrics-configuration '{"Id":"metrics-config-id", "Filter":
{"AccessPointArn":"arn:aws:s3:Region:account-id:accesspoint/access-point-name"}} '
```

Example: To filter by prefix, tags, and access point

```
aws s3api put-bucket-metrics-configuration --endpoint https://
s3.Region.amazonaws.com --bucket amzn-s3-demo-bucket --id metrics-config-id --
metrics-configuration '
{
    "Id": "metrics-config-id",
    "Filter": {
        "And": {
            "Prefix": "string",
```

```
            "Tags": [
                {
                    "Key": "string",
                    "Value": "string"
                }
            ],
            "AccessPointArn": "arn:aws:s3:Region:account-id:accesspoint/access-
point-name"
        }
    }
}'
```

Using the REST API

You can also add metrics configurations programmatically with the Amazon S3 REST API. For more information about adding and working with metrics configurations, see the following topics in the *Amazon Simple Storage Service API Reference*:

- PUT Bucket Metric Configuration
- GET Bucket Metric Configuration
- List Bucket Metric Configuration
- DELETE Bucket Metric Configuration

Deleting a metrics filter

You can delete an Amazon CloudWatch request metrics filter if you no longer need it. When you delete a filter, you stop being charged for request metrics that use that *specific filter*. However, charges still apply for any other filters that remain.

You cannot undo a filter deletion. After you delete a filter, you can no longer use it for request metrics.

To create a request metrics filter, see the following topics:

- Creating a CloudWatch metrics configuration for all the objects in your bucket
- Creating a metrics configuration that filters by prefix, object tag, or access point

Using the S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket you want to delete a request metrics filter for.
4. Choose the **Metrics** tab.
5. Under **Bucket metrics**, choose **View additional charts**.
6. Choose the **Request metrics** tab.
7. Choose **Manage filters**.
8. Choose your filter.

   > **Important**
   >
   > Deleting a filter cannot be undone.

9. Choose **Delete**.

   Amazon S3 deletes your filter.

Using the REST API

You can also add metrics configurations programmatically with the Amazon S3 REST API. For more information about adding and working with metrics configurations, see the following topics in the *Amazon Simple Storage Service API Reference*:

- PUT Bucket Metric Configuration
- GET Bucket Metric Configuration
- List Bucket Metric Configuration
- DELETE Bucket Metric Configuration

Amazon S3 Event Notifications

You can use the Amazon S3 Event Notifications feature to receive notifications when certain events happen in your S3 bucket. To enable notifications, add a notification configuration that

identifies the events that you want Amazon S3 to publish. Make sure that it also identifies the destinations where you want Amazon S3 to send the notifications. You store this configuration in the *notification* subresource that's associated with a bucket. For more information, see General purpose buckets configuration options. Amazon S3 provides an API for you to manage this subresource.

> **Important**
>
> Amazon S3 event notifications are designed to be delivered at least once. Typically, event notifications are delivered in seconds but can sometimes take a minute or longer.

Overview of Amazon S3 Event Notifications

Currently, Amazon S3 can publish notifications for the following events:

- New object created events
- Object removal events
- Restore object events
- Reduced Redundancy Storage (RRS) object lost events
- Replication events
- S3 Lifecycle expiration events
- S3 Lifecycle transition events
- S3 Intelligent-Tiering automatic archival events
- Object tagging events
- Object ACL PUT events
- Object Annotation events
- Object Retention events

For full descriptions of all the supported event types, see Supported event types for SQS, SNS, and Lambda.

Amazon S3 can send event notification messages to the following destinations. You specify the Amazon Resource Name (ARN) value of these destinations in the notification configuration.

- Amazon Simple Notification Service (Amazon SNS) topics

- Amazon Simple Queue Service (Amazon SQS) queues
- AWS Lambda function
- Amazon EventBridge

For more information, see Supported event destinations.

Note

Amazon Simple Queue Service FIFO (First-In-First-Out) queues aren't supported as an Amazon S3 event notification destination. To send a notification for an Amazon S3 event to an Amazon SQS FIFO queue, you can use Amazon EventBridge. For more information, see Enabling Amazon EventBridge.

Note

AWS Lambda durable functions aren't supported as an Amazon S3 event notification destination.

Warning

If your notification writes to the same bucket that triggers the notification, it could cause an execution loop. For example, if the bucket triggers a Lambda function each time an object is uploaded, and the function uploads an object to the bucket, then the function indirectly triggers itself. To avoid this, use two buckets, or configure the trigger to only apply to a prefix used for incoming objects.
For more information and an example of using Amazon S3 notifications with AWS Lambda, see Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

For more information about the number of event notification configurations that you can create per bucket, see Amazon S3 service quotas in *AWS General Reference*.

For more information about event notifications, see the following sections.

Topics

- Event notification types and destinations
- Using Amazon SQS, Amazon SNS, and Lambda
- Using EventBridge

Event notification types and destinations

Amazon S3 supports several event notification types and destinations where the notifications can be published. You can specify the event type and destination when configuring your event notifications. Only one destination can be specified for each event notification. Amazon S3 event notifications send one event entry for each notification message.

Topics

- Supported event destinations
- Supported event types for SQS, SNS, and Lambda
- Supported event types for Amazon EventBridge
- Event ordering and duplicate events

Supported event destinations

Amazon S3 can send event notification messages to the following destinations.

- Amazon Simple Notification Service (Amazon SNS) topics
- Amazon Simple Queue Service (Amazon SQS) queues
- AWS Lambda
- Amazon EventBridge

However, only one destination type can be specified for each event notification.

> **Note**
>
> You must grant Amazon S3 permissions to post messages to an Amazon SNS topic or an Amazon SQS queue. You must also grant Amazon S3 permission to invoke an AWS Lambda function on your behalf. For instructions on how to grant these permissions, see Granting permissions to publish event notification messages to a destination.

Amazon SNS topic

Amazon SNS is a flexible, fully managed push messaging service. You can use this service to push messages to mobile devices or distributed services. With SNS, you can publish a message once, and deliver it one or more times. Currently, Standard SNS is only allowed as an S3 event notification destination, whereas SNS FIFO is not allowed.

Amazon SNS both coordinates and manages sending and delivering messages to subscribing endpoints or clients. You can use the Amazon SNS console to create an Amazon SNS topic that your notifications can be sent to.

The topic must be in the same AWS Region as your Amazon S3 bucket. For instructions on how to create an Amazon SNS topic, see Getting started with Amazon SNS in the *Amazon Simple Notification Service Developer Guide* and the Amazon SNS FAQ.

Before you can use the Amazon SNS topic that you created as an event notification destination, you need the following:

- The Amazon Resource Name (ARN) for the Amazon SNS topic
- A valid Amazon SNS topic subscription. With it, topic subscribers are notified when a message is published to your Amazon SNS topic.

Amazon SQS queue

Amazon SQS offers reliable and scalable hosted queues for storing messages as they travel between computers. You can use Amazon SQS to transmit any volume of data without requiring other services to be always available. You can use the Amazon SQS console to create an Amazon SQS queue that your notifications can be sent to.

The Amazon SQS queue must be in the same AWS Region as your Amazon S3 bucket. For instructions on how to create an Amazon SQS queue, see What is Amazon Simple Queue Service and Getting started with Amazon SQS in the *Amazon Simple Queue Service Developer Guide*.

Before you can use the Amazon SQS queue as an event notification destination, you need the following:

- The Amazon Resource Name (ARN) for the Amazon SQS queue

Note

Amazon Simple Queue Service FIFO (First-In-First-Out) queues aren't supported as an Amazon S3 event notification destination. To send a notification for an Amazon S3 event to an Amazon SQS FIFO queue, you can use Amazon EventBridge. For more information, see Enabling Amazon EventBridge.

Amazon SQS message size

Amazon S3 event notification messages can be up to 64 KB. We recommend that you set the queue's `MaximumMessageSize` attribute to at least 64 KB. If an event notification exceeds the queue's configured `MaximumMessageSize`, Amazon SQS rejects the notification and doesn't deliver the event.

Lambda function

You can use AWS Lambda to extend other AWS services with custom logic, or create your own backend that operates at AWS scale, performance, and security. With Lambda, you can create discrete, event-driven applications that run only when needed. You can also use it to scale these applications automatically from a few requests a day to thousands a second.

Lambda can run custom code in response to Amazon S3 bucket events. You upload your custom code to Lambda and create what's called a Lambda function. When Amazon S3 detects an event of a specific type, it can publish the event to AWS Lambda and invoke your function in Lambda. In response, Lambda runs your function. One event type it might detect, for example, is an object created event.

You can use the AWS Lambda console to create a Lambda function that uses the AWS infrastructure to run the code on your behalf. The Lambda function must be in the same Region as your S3 bucket. You must also have the name or the ARN of a Lambda function to set up the Lambda function as an event notification destination.

Warning

If your notification writes to the same bucket that triggers the notification, it could cause an execution loop. For example, if the bucket triggers a Lambda function each time an

object is uploaded, and the function uploads an object to the bucket, then the function indirectly triggers itself. To avoid this, use two buckets, or configure the trigger to only apply to a prefix used for incoming objects.
For more information and an example of using Amazon S3 notifications with AWS Lambda, see Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

Amazon EventBridge

Amazon EventBridge is a serverless event bus, which receives events from AWS services. You can set up rules to match events and deliver them to targets, such as an AWS service or an HTTP endpoint. For more information, see What is EventBridge in the *Amazon EventBridge User Guide*.

Unlike other destinations, you can either enable or disable events to be delivered to EventBridge for a bucket. If you enable delivery, all events are sent to EventBridge. Moreover, you can use EventBridge rules to route events to additional targets.

Supported event types for SQS, SNS, and Lambda

Amazon S3 can publish events of the following types. You specify these event types in the notification configuration.

Event types	Description
`s3:TestEvent`	When a notification is enabled, Amazon S3 publishes a test notification. This is to ensure that the topic exists and that the bucket owner has permission to publish the specified topic. If enabling the notification fails, you don't receive a test notification.

`s3:ObjectCreated:*`
 	
`s3:ObjectCreated:Put`
 	
`s3:ObjectCreated:Post`
 	
`s3:ObjectCreated:Copy`
	Amazon S3 API operations such as PUT, POST, and COPY can create an object. With these event types, you can enable notifications when an object is created using a specific API operation. Alternatively, you can use the `s3:ObjectCreated:*` event type to request notification regardless of the API that was used to create an object.

Event types	Description
s3:ObjectCreated:CompleteMultipartUpload	s3:ObjectCreated:CompleteMultipartUpload includes objects that are created using UploadPartCopy for Copy operations.
s3:ObjectRemoved:* s3:ObjectRemoved:Delete s3:ObjectRemoved:DeleteMarkerCreated	By using the ObjectRemoved event types, you can enable notification when an object or a batch of objects is removed from a bucket. You can request notification when an object is deleted or a versioned object is permanently deleted by using the s3:ObjectRemoved:Delete event type. Alternatively, you can request notification when a delete marker is created for a versioned object using s3:ObjectRemoved:DeleteMarkerCreated. For instructions on how to delete versioned objects, see Deleting object versions from a versioning-enabled bucket. You can also use a wildcard s3:ObjectRemoved:* to request notification anytime an object is deleted. These event notifications don't alert you for automatic deletes from lifecycle configurations or from failed operations.

Event types	Description

`s3:ObjectRestore:*`
 	
`s3:ObjectRestore:Post`
 	
`s3:ObjectRestore:Completed`
 	
`s3:ObjectRestore:Delete`
	By using the `ObjectRestore` event types, you can receive notifications for event initiation and completion when restoring objects from the S3 Glacier Flexible Retrieval storage class, S3 Glacier Deep Archive storage class, S3 Intelligent-Tiering Archive Access tier, and S3 Intelligent-Tiering Deep Archive Access tier. You can also receive notifications for when the restored copy of an object expires.
	
	The `s3:ObjectRestore:Post` event type notifies you of object restoration initiation. The `s3:ObjectRestore:Completed` event type notifies you of restoration completion. The `s3:ObjectRestore:Delete` event type notifies you when the temporary copy of a restored object expires.

`s3:ReducedRedundancyLostObject`	You receive this notification event when Amazon S3 detects that an object of the RRS storage class is lost.

Event types	Description

`s3:Replication:*`
 	
`s3:Replication:OperationFailedReplication`
 	
`s3:Replication:OperationMissedThreshold`
 	
`s3:Replication:OperationReplicatedAfterThreshold`
 	
`s3:Replication:OperationNotTracked`
	By using the `Replication` event types, you can receive notifications for replication configurations that have S3 Replication metrics or S3 Replication Time Control (S3 RTC) enabled. You can monitor the minute-by-minute progress of replication events by tracking bytes pending, operations pending, and replication latency. For information about replication metrics, see Monitoring replication with metrics, event notifications, and statuses.
	
	• The `s3:Replication:OperationFailedReplication` event type notifies you when an object that was eligible for replication failed to replicate.
	• The `s3:Replication:OperationMissedThreshold` event type notifies you when an object that was eligible for replication that uses S3 RTC exceeds the 15-minute threshold for replication.
	• The `s3:Replication:OperationReplicatedAfterThreshold` event type notifies you when an object that was eligible for replication that uses S3 RTC replicates after the 15-minute threshold.
	• The `s3:Replication:OperationNotTracked` event type notifies you when an object that was eligible for live replication (either Same-Region Replication [SRR] or Cross-Region Replication [CRR]) is no longer being tracked by replication metrics.

Event types	Description
s3:LifecycleExpiration:* s3:LifecycleExpiration:Delete s3:LifecycleExpiration:DeleteMarkerCreated	By using the LifecycleExpiration event types, you can receive a notification when Amazon S3 deletes an object based on your S3 Lifecycle configuration. The s3:LifecycleExpiration:Delete event type notifies you when an object in an unversioned bucket is deleted. It also notifies you when an object version is permanently deleted by an S3 Lifecycle configuration. The s3:LifecycleExpiration:DeleteMarkerCreated event type notifies you when S3 Lifecycle creates a delete marker when a current version of an object in versioned bucket is deleted.
s3:LifecycleTransition	You receive this notification event when an object is transitioned to another Amazon S3 storage class by an S3 Lifecycle configuration.
s3:IntelligentTiering	You receive this notification event when an object within the S3 Intelligent-Tiering storage class moved to the Archive Access tier or Deep Archive Access tier.
s3:ObjectTagging:* s3:ObjectTagging:Put s3:ObjectTagging:Delete	By using the ObjectTagging event types, you can enable notification when an object tag is added or deleted from an object. The s3:ObjectTagging:Put event type notifies you when a tag is PUT on an object or an existing tag is updated. The s3:ObjectTagging:Delete event type notifies you when a tag is removed from an object.

Event types	Description

`s3:ObjectAnnotation:*`
 	
`s3:ObjectAnnotation:Put`
 	
`s3:ObjectAnnotation:Delete`
	By using the `ObjectAnnotation` event types, you can enable notification when an annotation is created, updated, or deleted on an object.
	
	The `s3:ObjectAnnotation:Put` event type notifies you when an annotation is created or updated through `PutObjectAnnotation`. The `s3:ObjectAnnotation:Delete` event type notifies you when an annotation is deleted through `DeleteObjectAnnotation`.

`s3:ObjectAcl:Put`	You receive this notification event when an ACL is PUT on an object or when an existing ACL is changed. An event is not generated when a request results in no change to an object's ACL.
`s3:ObjectRetention:Put`	You receive this notification event when you set a retention configuration on an object using `PutObjectRetention`. This includes setting fixed or variable retention, releasing an event hold, modifying the event hold duration, and changing the retain-until-date.

Note

Note the following behavior for annotation event notifications:

- `PutObjectAnnotation` and `DeleteObjectAnnotation` send only annotation-specific events. They do not send object-level events such as `s3:ObjectCreated:*`.
- `GetObjectAnnotation` and `ListObjectAnnotations` do not send event notifications.
- `CopyObject` sends the standard `s3:ObjectCreated:Copy` event. If the copied object has annotations, the event includes a `hasObjectAnnotation` field set to `true`.
- `DeleteObject` does not include annotation-related fields in the event payload.
- Prefix and suffix filters apply to object keys only, not to annotation names.

Supported event types for Amazon EventBridge

For a list of event types Amazon S3 will send to Amazon EventBridge, see Using EventBridge.

Event ordering and duplicate events

Amazon S3 Event Notifications is designed to deliver notifications at least once, but they aren't guaranteed to arrive in the same order that the events occurred. On rare occasions, Amazon S3's retry mechanism might cause duplicate S3 Event Notifications for the same object event. For more about handling duplicate or out of order events, see Manage event ordering and duplicate events with Amazon S3 Event Notifications on the *AWS Storage Blog*.

Using Amazon SQS, Amazon SNS, and Lambda

Enabling notifications is a bucket-level operation. You store notification configuration information in the *notification* subresource that's associated with a bucket. After you create or change the bucket notification configuration, it usually takes about five minutes for the changes to take effect. When the notification is first enabled, an `s3:TestEvent` occurs. You can use any of the following methods to manage notification configuration:

- **Using the Amazon S3 console** — You can use the console UI to set a notification configuration on a bucket without having to write any code. For more information, see Enabling and configuring event notifications using the Amazon S3 console.
- **Programmatically using the AWS SDKs** — Internally, both the console and the SDKs call the Amazon S3 REST API to manage *notification* subresources that are associated with the bucket. For examples of notification configurations that use AWS SDK, see Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue).

> **Note**
>
> You can also make the Amazon S3 REST API calls directly from your code. However, this can be cumbersome because to do so you must write code to authenticate your requests.

Regardless of the method that you use, Amazon S3 stores the notification configuration as XML in the *notification* subresource that's associated with a bucket. For information about bucket subresources, see General purpose buckets configuration options.

> **Note**
>
> If you have multiple failed event notifications due to deleted destinations you may receive the **Unable to validate the following destination configurations** when trying to delete them. You can resolve this in the S3 console by deleting all the failed notifications at the same time.

Topics

- Granting permissions to publish event notification messages to a destination
- Enabling and configuring event notifications using the Amazon S3 console
- Configuring event notifications programmatically
- Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue)
- Configuring event notifications using object key name filtering
- Event message structure

Granting permissions to publish event notification messages to a destination

You must grant the Amazon S3 principal the necessary permissions to call the relevant API to publish messages to an SNS topic, an SQS queue, or a Lambda function. This is so that Amazon S3 can publish event notification messages to a destination.

To troubleshoot publishing event notification messages to a destination, see Troubleshoot to publish Amazon S3 event notifications to an Amazon Simple Notification Service topic .

Topics

- Granting permissions to invoke an AWS Lambda function
- Granting permissions to publish messages to an SNS topic or an SQS queue

Granting permissions to invoke an AWS Lambda function

Amazon S3 publishes event messages to AWS Lambda by invoking a Lambda function and providing the event message as an argument.

When you use the Amazon S3 console to configure event notifications on an Amazon S3 bucket for a Lambda function, the console sets up the necessary permissions on the Lambda function. This is

so that Amazon S3 has permissions to invoke the function from the bucket. For more information, see Enabling and configuring event notifications using the Amazon S3 console.

You can also grant Amazon S3 permissions from AWS Lambda to invoke your Lambda function. For more information, see Tutorial: Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

Granting permissions to publish messages to an SNS topic or an SQS queue

To grant Amazon S3 permissions to publish messages to the SNS topic or SQS queue, attach an AWS Identity and Access Management (IAM) policy to the destination SNS topic or SQS queue.

For an example of how to attach a policy to an SNS topic or an SQS queue, see Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue). For more information about permissions, see the following topics:

- Example cases for Amazon SNS access control in the *Amazon Simple Notification Service Developer Guide*
- Identity and access management in Amazon SQS in the *Amazon Simple Queue Service Developer Guide*

IAM policy for a destination SNS topic

The following is an example of an AWS Identity and Access Management (IAM) policy that you attach to the destination SNS topic. For instructions on how to use this policy to set up a destination Amazon SNS topic for event notifications, see Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue).

JSON

```
{
    "Version":"2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "Example SNS topic policy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
```

```
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "arn:aws:sns:us-east-1:111122223333:example-sns-topic",
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:s3:::amzn-s3-demo-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-123456789012"
                }
            }
        }
    ]
}
```

IAM policy for a destination SQS queue

The following is an example of an IAM policy that you attach to the destination SQS queue. For instructions on how to use this policy to set up a destination Amazon SQS queue for event notifications, see Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue).

To use this policy, you must update the Amazon SQS queue ARN, bucket name, and bucket owner's AWS account ID.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SQS:SendMessage"
            ],
```

```
            "Resource": "arn:aws:sqs:us-east-1:111122223333:queue-name",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-123456789012"
                }
            }
        }
    ]
}
```

For both the Amazon SNS and Amazon SQS IAM policies, you can specify the `StringLike` condition in the policy instead of the `ArnLike` condition.

When `ArnLike` is used, the partition, service, account ID, resource type, and partial resource ID portions of the ARN must exactly match to the ARN in the request context. Only the Region and resource path allow partial matching.

When `StringLike` is used instead of `ArnLike`, matching ignores the ARN structure and allows partial matching, regardless of the portion that's replaced by the wildcard character. For more information, see IAM JSON policy elements in the *IAM User Guide*.

```
"Condition": {
  "StringLike": { "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket" }
  }
```

AWS KMS key policy

If the SQS queue or SNS topics are encrypted with an AWS Key Management Service (AWS KMS) customer managed key, you must grant the Amazon S3 service principal permission to work with the encrypted topics or queue. To grant the Amazon S3 service principal permission, add the following statement to the key policy for the customer managed key.

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information about AWS KMS key policies, see Using key policies in AWS KMS in the *AWS Key Management Service Developer Guide*.

For more information about using server-side encryption with AWS KMS for Amazon SQS and Amazon SNS, see the following:

- Key management in the *Amazon Simple Notification Service Developer Guide*.
- Key management in the *Amazon Simple Queue Service Developer Guide*.
- Encrypting messages published to Amazon SNS with AWS KMS in the *AWS Compute Blog*.

Enabling and configuring event notifications using the Amazon S3 console

You can enable certain Amazon S3 general purpose bucket events to send a notification message to a destination whenever those events occur. This section explains how to use the Amazon S3 console to enable event notifications. For information about how to use event notifications with the AWS SDKs and the Amazon S3 REST APIs, see Configuring event notifications programmatically.

Prerequisites: Before you can enable event notifications for your bucket, you must set up one of the destination types and then configure permissions. For more information, see Supported event destinations and Granting permissions to publish event notification messages to a destination.

Note

Amazon Simple Queue Service FIFO (First-In-First-Out) queues aren't supported as an Amazon S3 event notification destination. To send a notification for an Amazon S3 event to an Amazon SQS FIFO queue, you can use Amazon EventBridge. For more information, see Enabling Amazon EventBridge.

Topics

- Enabling Amazon SNS, Amazon SQS, or Lambda notifications using the Amazon S3 console

Enabling Amazon SNS, Amazon SQS, or Lambda notifications using the Amazon S3 console

To enable and configure event notifications for an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to enable events for.
4. Choose **Properties**.
5. Navigate to the **Event Notifications** section and choose **Create event notification**.
6. In the **General configuration** section, specify descriptive event name for your event notification. Optionally, you can also specify a prefix and a suffix to limit the notifications to objects with keys ending in the specified characters.

   a. Enter a description for the **Event name**.

      If you don't enter a name, a globally unique identifier (GUID) is generated and used for the name.

   b. (Optional) To filter event notifications by prefix, enter a **Prefix**.

      For example, you can set up a prefix filter so that you receive notifications only when files are added to a specific folder (for example, `images/`).

   c. (Optional) To filter event notifications by suffix, enter a **Suffix**.

      For more information, see Configuring event notifications using object key name filtering.

7. In the **Event types** section, select one or more event types that you want to receive notifications for.

   For a list of the different event types, see Supported event types for SQS, SNS, and Lambda.

8. In the **Destination** section, choose the event notification destination.

   > **Note**
   >
   > Before you can publish event notifications, you must grant the Amazon S3 principal the necessary permissions to call the relevant API. This is so that it can publish notifications to a Lambda function, SNS topic, or SQS queue.

   a. Select the destination type: **Lambda Function**, **SNS Topic**, or **SQS Queue**.

   b. After you choose your destination type, choose a function, topic, or queue from the list.

   c. Or, if you prefer to specify an Amazon Resource Name (ARN), select **Enter ARN** and enter the ARN.

   For more information, see Supported event destinations.

9. Choose **Save changes**, and Amazon S3 sends a test message to the event notification destination.

Configuring event notifications programmatically

By default, notifications aren't enabled for any type of event. Therefore, the *notification* subresource initially stores an empty configuration.

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
</NotificationConfiguration>
```

To enable notifications for events of specific types, you replace the XML with the appropriate configuration that identifies the event types you want Amazon S3 to publish and the destination where you want the events published. For each destination, you add a corresponding XML configuration.

To publish event messages to an SQS queue

To set an SQS queue as the notification destination for one or more event types, add the `QueueConfiguration`.

```
<NotificationConfiguration>
  <QueueConfiguration>
    <Id>optional-id-string</Id>
    <Queue>sqs-queue-arn</Queue>
    <Event>event-type</Event>
    <Event>event-type</Event>
     ...
  </QueueConfiguration>
   ...
</NotificationConfiguration>
```

To publish event messages to an SNS topic

To set an SNS topic as the notification destination for specific event types, add the `TopicConfiguration`.

```
<NotificationConfiguration>
  <TopicConfiguration>
     <Id>optional-id-string</Id>
     <Topic>sns-topic-arn</Topic>
     <Event>event-type</Event>
     <Event>event-type</Event>
      ...
  </TopicConfiguration>
   ...
</NotificationConfiguration>
```

To invoke the AWS Lambda function and provide an event message as an argument

To set a Lambda function as the notification destination for specific event types, add the `CloudFunctionConfiguration`.

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>optional-id-string</Id>
     <CloudFunction>cloud-function-arn</CloudFunction>
     <Event>event-type</Event>
     <Event>event-type</Event>
      ...
```

```
  </CloudFunctionConfiguration>
   ...
</NotificationConfiguration>
```

To remove all notifications configured on a bucket

To remove all notifications configured on a bucket, save an empty `<NotificationConfiguration/>` element in the *notification* subresource.

When Amazon S3 detects an event of the specific type, it publishes a message with the event information. For more information, see Event message structure.

For more information about configuring event notifications, see the following topics:

- Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue).
- Configuring event notifications using object key name filtering

Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue)

You can receive Amazon S3 notifications using Amazon Simple Notification Service (Amazon SNS) or Amazon Simple Queue Service (Amazon SQS). In this walkthrough, you add a notification configuration to your bucket using an Amazon SNS topic and an Amazon SQS queue.

> **Note**
>
> Amazon Simple Queue Service FIFO (First-In-First-Out) queues aren't supported as an Amazon S3 event notification destination. To send a notification for an Amazon S3 event to an Amazon SQS FIFO queue, you can use Amazon EventBridge. For more information, see Enabling Amazon EventBridge.

Topics

- Walkthrough summary
- Step 1: Create an Amazon SQS queue
- Step 2: Create an Amazon SNS topic
- Step 3: Add a notification configuration to your bucket
- Step 4: Test the setup

Walkthrough summary

This walkthrough helps you do the following:

- Publish events of the `s3:ObjectCreated:*` type to an Amazon SQS queue.
- Publish events of the `s3:ReducedRedundancyLostObject` type to an Amazon SNS topic.

For information about notification configuration, see Using Amazon SQS, Amazon SNS, and Lambda.

You can do all these steps using the console, without writing any code. In addition, code examples using AWS SDKs for Java and .NET are also provided to help you add notification configurations programmatically.

The procedure includes the following steps:

1. Create an Amazon SQS queue.

   Using the Amazon SQS console, create an SQS queue. You can access any messages Amazon S3 sends to the queue programmatically. But, for this walkthrough, you verify notification messages in the console.

   You attach an access policy to the queue to grant Amazon S3 permission to post messages.
2. Create an Amazon SNS topic.

   Using the Amazon SNS console, create an SNS topic and subscribe to the topic. That way, any events posted to it are delivered to you. You specify email as the communications protocol. After you create a topic, Amazon SNS sends an email. You use the link in the email to confirm the topic subscription.

   You attach an access policy to the topic to grant Amazon S3 permission to post messages.
3. Add notification configuration to a bucket.

Step 1: Create an Amazon SQS queue

Follow the steps to create and subscribe to an Amazon Simple Queue Service (Amazon SQS) queue.

1. Using the Amazon SQS console, create a queue. For instructions, see Getting Started with Amazon SQS in the *Amazon Simple Queue Service Developer Guide*.

2. Replace the access policy that's attached to the queue with the following policy.

   a. In the Amazon SQS console, in the **Queues** list, choose the queue name.

   b. On the **Access policy** tab, choose **Edit**.

   c. Replace the access policy that's attached to the queue. In it, provide your Amazon SQS ARN, source bucket name, and bucket owner account ID.

      JSON

```
{
    "Version":"2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SQS:SendMessage"
            ],
            "Resource": "arn:aws:sqs:us-west-2:111122223333:s3-
notification-queue",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-123456789012"
                }
            }
        }
    ]
}
```

   d. Choose **Save**.

3. (Optional) If the Amazon SQS queue or the Amazon SNS topic is server-side encryption enabled with AWS Key Management Service (AWS KMS), add the following policy to the associated symmetric encryption customer managed key.

You must add the policy to a customer managed key because you cannot modify the AWS managed key for Amazon SQS or Amazon SNS.

JSON

```
{
    "Version":"2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
    ]
}
```

For more information about using SSE for Amazon SQS and Amazon SNS with AWS KMS, see the following:

- Key management in the *Amazon Simple Notification Service Developer Guide*.
- Key management in the *Amazon Simple Queue Service Developer Guide*.

4. Note the queue ARN.

   The SQS queue that you created is another resource in your AWS account. It has a unique Amazon Resource Name (ARN). You need this ARN in the next step. The ARN is of the following format:

   ```
   arn:aws:sqs:aws-region:account-id:queue-name
   ```

Step 2: Create an Amazon SNS topic

Follow the steps to create and subscribe to an Amazon SNS topic.

1. Using Amazon SNS console, create a topic. For instructions, see Creating an Amazon SNS topic in the *Amazon Simple Notification Service Developer Guide*.
2. Subscribe to the topic. For this exercise, use email as the communications protocol. For instructions, see Subscribing to an Amazon SNS topic in the *Amazon Simple Notification Service Developer Guide*.

   You get an email requesting you to confirm your subscription to the topic. Confirm the subscription.
3. Replace the access policy attached to the topic with the following policy. In it, provide your SNS topic ARN, bucket name, and bucket owner's account ID.

```
{
    "Version": "2012-10-17",
    "Id": "example-ID",
    "Statement": [
        {
            "Sid": "example-statement-ID",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
            "Action": [
                "SNS:Publish"
            ],
            "Resource": "arn:aws:sns:us-west-2:111122223333:s3-notification-topic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:*:*:awsexamplebucket1"
                },
                "StringEquals": {
                    "aws:SourceAccount": "bucket-owner-123456789012"
                }
            }
        }
    ]
}
```

4. Note the topic ARN.

The SNS topic you created is another resource in your AWS account, and it has a unique ARN. You will need this ARN in the next step. The ARN will be of the following format:

```
arn:aws:sns:aws-region:account-id:topic-name
```

Step 3: Add a notification configuration to your bucket

You can enable bucket notifications either by using the Amazon S3 console or programmatically by using AWS SDKs. Choose any one of the options to configure notifications on your bucket. This section provides code examples using the AWS SDKs for Java and .NET.

Option A: Enable notifications on a bucket using the console

Using the Amazon S3 console, add a notification configuration requesting Amazon S3 to do the following:

- Publish events of the **All object create events** type to your Amazon SQS queue.
- Publish events of the **Object in RRS lost** type to your Amazon SNS topic.

After you save the notification configuration, Amazon S3 posts a test message, which you get through email.

For instructions, see Enabling and configuring event notifications using the Amazon S3 console.

Option B: Enable notifications on a bucket using the AWS SDKs

.NET

The following C# code example provides a complete code listing that adds a notification configuration to a bucket. You must update the code and provide your bucket name and SNS topic ARN. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
```

```
namespace Amazon.DocSamples.S3
{
    class EnableNotificationsTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string snsTopic = "*** SNS topic ARN ***";
        private const string sqsQueue = "*** SQS topic ARN ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            EnableNotificationAsync().Wait();
        }

        static async Task EnableNotificationAsync()
        {
            try
            {
               PutBucketNotificationRequest request = new
 PutBucketNotificationRequest
                {
                    BucketName = bucketName
                };

                TopicConfiguration c = new TopicConfiguration
                {
                    Events = new List<EventType> { EventType.ObjectCreatedCopy },
                    Topic = snsTopic
                };
                request.TopicConfigurations = new List<TopicConfiguration>();
                request.TopicConfigurations.Add(c);
                request.QueueConfigurations = new List<QueueConfiguration>();
                request.QueueConfigurations.Add(new QueueConfiguration()
                {
                    Events = new List<EventType> { EventType.ObjectCreatedPut },
                    Queue = sqsQueue
                });
```

```
                PutBucketNotificationResponse response = await
 client.PutBucketNotificationAsync(request);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' ",
 e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown error encountered on server.
 Message:'{0}' ", e.Message);
            }
        }
    }
}
```

Java

For examples of how to configure bucket notifications with the AWS SDK for Java, see Process S3 event notifications in the *Amazon S3 API Reference*.

Step 4: Test the setup

Now, you can test the setup by uploading an object to your bucket and verifying the event notification in the Amazon SQS console. For instructions, see Receiving a Message in the *Amazon Simple Queue Service Developer Guide "Getting Started" section*.

Configuring event notifications using object key name filtering

When configuring an Amazon S3 event notification, you must specify which supported Amazon S3 event types cause Amazon S3 to send the notification. If an event type that you didn't specify occurs in your S3 bucket, Amazon S3 doesn't send the notification.

You can configure notifications to be filtered by the prefix and suffix of the key name of objects. For example, you can set up a configuration where you're sent a notification only when image files with a "`.jpg`" file name extension are added to a bucket. Or, you can have a configuration that delivers a notification to an Amazon SNS topic when an object with the prefix "`images/`" is added to the bucket, while having notifications for objects with a "`logs/`" prefix in the same bucket delivered to an AWS Lambda function.

Note

A wildcard character ("*") can't be used in filters as a prefix or suffix. If your prefix or suffix contains a space, you must replace it with the "+" character. If you use any other special characters in the value of the prefix or suffix, you must enter them in URL-encoded (percent-encoded) format. For a complete list of special characters that must be converted to URL-encoded format when used in a prefix or suffix for event notifications, see Safe characters.

You can set up notification configurations that use object key name filtering in the Amazon S3 console. You can do so by using Amazon S3 APIs through the AWS SDKs or the REST APIs directly. For information about using the console UI to set a notification configuration on a bucket, see Enabling and configuring event notifications using the Amazon S3 console.

Amazon S3 stores the notification configuration as XML in the *notification* subresource associated with a bucket as described in Using Amazon SQS, Amazon SNS, and Lambda. You use the `Filter` XML structure to define the rules for notifications to be filtered by the prefix or suffix of an object key name. For information about the `Filter` XML structure, see PUT Bucket notification in the *Amazon Simple Storage Service API Reference*.

Notification configurations that use `Filter` cannot define filtering rules with overlapping prefixes, overlapping suffixes, or prefix and suffix overlapping. The following sections have examples of valid notification configurations with object key name filtering. They also contain examples of notification configurations that are not valid because of prefix and suffix overlapping.

Topics

- Examples of valid notification configurations with object key name filtering
- Examples of notification configurations with invalid prefix and suffix overlapping

Examples of valid notification configurations with object key name filtering

The following notification configuration contains a queue configuration identifying an Amazon SQS queue for Amazon S3 to publish events to of the `s3:ObjectCreated:Put` type. The events are published whenever an object that has a prefix of `images/` and a `jpg` suffix is PUT to a bucket.

```
<NotificationConfiguration>
  <QueueConfiguration>
```

```
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>images/</Value>
              </FilterRule>
              <FilterRule>
                  <Name>suffix</Name>
                  <Value>jpg</Value>
              </FilterRule>
          </S3Key>
      </Filter>
      <Queue>arn:aws:sqs:us-west-2:444455556666:s3notificationqueue</Queue>
      <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

The following notification configuration has multiple non-overlapping prefixes. The configuration defines that notifications for PUT requests in the `images/` folder go to queue-A, while notifications for PUT requests in the `logs/` folder go to queue-B.

```
<NotificationConfiguration>
  <QueueConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images/</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-A</Queue>
     <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
  <QueueConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>logs/</Value>
```

```
                </FilterRule>
            </S3Key>
     </Filter>
     <Queue>arn:aws:sqs:us-west-2:444455556666:sqs-queue-B</Queue>
     <Event>s3:ObjectCreated:Put</Event>
  </QueueConfiguration>
</NotificationConfiguration>
```

The following notification configuration has multiple non-overlapping suffixes. The configuration defines that all `.jpg` images newly added to the bucket are processed by Lambda cloud-function-A, and all newly added `.png` images are processed by cloud-function-B. The `.png` and `.jpg` suffixes aren't overlapping even though they have the same last letter. If a given string can end with both suffixes, the two suffixes are considered overlapping. A string can't end with both `.png` and `.jpg`, so the suffixes in the example configuration aren't overlapping suffixes.

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.png</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
```

```
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

Your notification configurations that use `Filter` can't define filtering rules with overlapping prefixes for the same event types. They can only do so, if the overlapping prefixes that are used with suffixes that don't overlap. The following example configuration shows how objects created with a common prefix but non-overlapping suffixes can be delivered to different destinations.

```
<NotificationConfiguration>
  <CloudFunctionConfiguration>
     <Id>1</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.jpg</Value>
                </FilterRule>
            </S3Key>
     </Filter>
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-A</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
  <CloudFunctionConfiguration>
     <Id>2</Id>
     <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>.png</Value>
                </FilterRule>
            </S3Key>
     </Filter>
```

```
     <CloudFunction>arn:aws:lambda:us-west-2:444455556666:cloud-function-B</
CloudFunction>
     <Event>s3:ObjectCreated:Put</Event>
  </CloudFunctionConfiguration>
</NotificationConfiguration>
```

Examples of notification configurations with invalid prefix and suffix overlapping

For the most part, your notification configurations that use `Filter` can't define filtering rules with overlapping prefixes, overlapping suffixes, or overlapping combinations of prefixes and suffixes for the same event types. You can have overlapping prefixes as long as the suffixes don't overlap. For an example, see Configuring event notifications using object key name filtering.

You can use overlapping object key name filters with different event types. For example, you can create a notification configuration that uses the prefix `image/` for the `ObjectCreated:Put` event type and the prefix `image/` for the `ObjectRemoved:*` event type.

You get an error if you try to save a notification configuration that has invalid overlapping name filters for the same event types when using the Amazon S3 console or API. This section shows examples of notification configurations that aren't valid because of overlapping name filters.

Any existing notification configuration rule is assumed to have a default prefix and suffix that match any other prefix and suffix, respectively. The following notification configuration isn't valid because it has overlapping prefixes. Specifically, the root prefix overlaps with any other prefix. The same thing is true if you use a suffix instead of a prefix in this example. The root suffix overlaps with any other suffix.

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-notification-two</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>prefix</Name>
                     <Value>images</Value>
                 </FilterRule>
            </S3Key>
```

```
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration>
```

The following notification configuration isn't valid because it has overlapping suffixes. If a given string can end with both suffixes, the two suffixes are considered overlapping. A string can end with `jpg` and `pg`. So, the suffixes overlap. The same is true for prefixes. If a given string can begin with both prefixes, the two prefixes are considered overlapping.

```
 <NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>pg</Value>
                 </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
</NotificationConfiguration
```

The following notification configuration isn't valid because it has overlapping prefixes and suffixes.

```
<NotificationConfiguration>
     <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-one</Topic>
         <Event>s3:ObjectCreated:*</Event>
```

```
        <Filter>
            <S3Key>
                <FilterRule>
                    <Name>prefix</Name>
                    <Value>images</Value>
                </FilterRule>
                <FilterRule>
                    <Name>suffix</Name>
                    <Value>jpg</Value>
                </FilterRule>
            </S3Key>
        </Filter>
    </TopicConfiguration>
    <TopicConfiguration>
         <Topic>arn:aws:sns:us-west-2:444455556666:sns-topic-two</Topic>
         <Event>s3:ObjectCreated:Put</Event>
         <Filter>
             <S3Key>
                 <FilterRule>
                     <Name>suffix</Name>
                     <Value>jpg</Value>
                 </FilterRule>
             </S3Key>
         </Filter>
    </TopicConfiguration>
</NotificationConfiguration>
```

Event message structure

The notification message that Amazon S3 sends to publish an event is in the JSON format.

For a general overview and instructions on configuring event notifications, see Amazon S3 Event Notifications.

This example shows *version 2.6* of the event notification JSON structure. Previously, Amazon S3 used different versions for different event types. General events used version 2.1, cross-Region replication events used 2.2, and S3 Lifecycle, S3 Intelligent-Tiering, object ACL, object tagging, and object restoration delete events used version 2.3.

Starting with version 2.4, Amazon S3 uses a single unified version for all event types, and the version increments consistently across all event types whenever the schema evolves. Each event contains extra information specific to the operation.

The maximum size of an event notification message is 64 KB. When using Amazon SQS as a destination, we recommend that you set your queue's `MaximumMessageSize` attribute to at least 64 KB. If an event notification exceeds the queue's configured `MaximumMessageSize`, Amazon SQS rejects the notification and doesn't deliver the event. For more information, see Event notification types and destinations.

```
{
   "Records":[
      {
         "eventVersion":"2.6",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"The time, in ISO-8601 format (for example,
 1970-01-01T00:00:00.000Z) when Amazon S3 finished processing the request",
         "eventName":"The event type",
         "userIdentity":{
            "principalId":"The unique ID of the IAM resource that caused the event"
         },
         "requestParameters":{
            "sourceIPAddress":"The IP address where the request came from"
         },
         "responseElements":{
            "x-amz-request-id":"The Amazon S3 generated request ID",
            "x-amz-id-2":"The Amazon S3 host that processed the request"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"The ID found in the bucket notification configuration",
            "bucket":{
               "name":"The name of the bucket, for example, amzn-s3-demo-bucket",
               "ownerIdentity":{
                  "principalId":"The Amazon retail customer ID of the bucket owner"
               },
               "arn":"The bucket Amazon Resource Name (ARN)",
               "awsGeneratedTags":"A map of tag key-value pairs generated by AWS
 services and attached to the bucket; only present when AWS generated tags exist on the
 bucket. This field is not present for cross-Region replication events"
            },
            "object":{
               "key":"The object key name",
               "size":"The object size in bytes (as a number)",
               "eTag":"The object entity tag (ETag)",
```

```
                "versionId":"The object version if the bucket is versioning-enabled;
 null or not present if the bucket isn't versioning-enabled",
                "sequencer": "A string representation of a hexadecimal value used to
 determine event sequence; only used with PUT and DELETE requests"
             }
          },
          "glacierEventData": {
             "restoreEventData": {
                "lifecycleRestorationExpiryTime": "The time, in ISO-8601 format (for
 example, 1970-01-01T00:00:00.000Z), when the temporary copy of the restored object
 expires",
                "lifecycleRestoreStorageClass": "The source storage class for restored
 objects"
             }
          },
          "objectRetentionEventData": {
             "mode": "The Object Lock retention mode applied to the object: COMPLIANCE
 or GOVERNANCE",
             "retainUntilDate": "The date, in ISO-8601 format, until which the object is
 retained",
             "eventHold": "The event hold state after the change: ON or OFF; present for
 event-hold operations",
             "eventHoldDuration": "The event hold duration after the change, as {"days":
 N} or {"years": N}. Present when an event hold duration is set or modified."
          }
       }
    ]
}
```

Note the following about the event message structure:

- The `eventVersion` key value contains a major and minor version in the form *`major.minor`*.

  The major version is incremented if Amazon S3 makes a change to the event structure that's not backward compatible. This includes removing a JSON field that's already present or changing how the contents of a field are represented (for example, a date format).

  The minor version is incremented if Amazon S3 makes a backward-compatible change to the event structure. This includes adding new fields to the event structure or introducing new event types. To stay compatible with new minor versions of the event structure, we recommend that your applications ignore new fields.

To ensure that your applications can parse the event structure correctly, we recommend that you do an equal-to comparison on the major version number. To ensure that the fields that are expected by your application are present, we also recommend doing a greater-than-or-equal-to comparison on the minor version.

- The `eventName` key value references the list of event notification types but doesn't contain the `s3:` prefix.
- The `userIdentity` key value references the unique ID of the AWS Identity and Access Management (IAM) resource (a user, role, group, and so on) that caused the event. For a definition of each IAM identification prefix (for example, AIDA, AROA, AGPA) and information about how to get the unique identifier, see Unique identifiers in the *IAM User Guide*.
- The `responseElements` key value is useful if you want to trace a request by following up with AWS Support. Both `x-amz-request-id` and `x-amz-id-2` help Amazon S3 trace an individual request. These values are the same as those that Amazon S3 returns in the response to the request that initiates the events. Therefore, you can use these values to match the event to the request.
- The `s3` key value provides information about the bucket and object involved in the event. The object key name value is URL encoded. For example, `red flower.jpg` becomes `red +flower.jpg`. (Amazon S3 returns "`application/x-www-form-urlencoded`" as the content type in the response.)

  The `ownerIdentity` key value corresponds to the Amazon retail (Amazon.com) customer ID of the bucket owner. This ID value is no longer used and is maintained only for backward compatibility.
- The `sequencer` key value provides a way to determine the sequence of events. Event notifications aren't guaranteed to arrive in the same order that the events occurred. However, notifications from events that create objects (PUT requests) and delete objects contain a `sequencer`. You can use this value to determine the order of events for a given object key.

  If you compare the `sequencer` strings from two event notifications on the same object key, the event notification with the greater `sequencer` hexadecimal value is the event that occurred later. If you're using event notifications to maintain a separate database or index of your Amazon S3 objects, we recommend that you compare and store the `sequencer` values as you process each event notification.

  Note the following:

  - You can't use the `sequencer` key value to determine the order for events on different object keys.
  - The `sequencer` strings can be of different lengths. So, to compare these values, first left-pad the shorter value with zeros, and then do a lexicographical comparison.
- The `glacierEventData` key value is only visible for `s3:ObjectRestore:Completed` events.
- The `restoreEventData` key value contains attributes that are related to your restore request.
- The `replicationEventData` key value is only visible for replication events.
- The `intelligentTieringEventData` key value is only visible for S3 Intelligent-Tiering events.
- The `lifecycleEventData` key value is only visible for S3 Lifecycle transition events.
- The `objectAnnotation` key value is only visible for annotation events (`ObjectAnnotation:Put` and `ObjectAnnotation:Delete`). It contains an array with the annotation `name`, `size` (Put events only), and `eTag` (Put events only).
- For `ObjectCreated:Copy` events, the `object` block includes a `hasObjectAnnotation` boolean field that indicates whether the copied object has annotations.
- The `objectRetentionEventData` key value is only visible for Object Lock retention events (`ObjectRetention:Put`). It contains the object's Object Lock retention state after the change. The `mode` and `retainUntilDate` fields are always present. The `eventHold` field (`ON` or `OFF`) is present for event-hold operations, and `eventHoldDuration` is present when an event hold duration is set or modified.

Example messages

The following are examples of Amazon S3 event notification messages.

Amazon S3 test message

After you configure an event notification on a bucket, Amazon S3 sends the following test message.

```
{
   "Service":"Amazon S3",
   "Event":"s3:TestEvent",
   "Time":"2014-10-13T15:57:02.089Z",
   "Bucket":"amzn-s3-demo-bucket",
   "RequestId":"5582815E1AEA5ADF",
   "HostId":"8cLeGAmw098X5cv4Zkwcmo8vvZa3eH3eKxsPzbB9wrR+YstdA6Knx4Ip8EXAMPLE"
```

```
}
```

> **Note**
>
> The `s3:TestEvent` message uses a different format than regular S3 event notifications. Unlike other event notifications that use the `Records` array structure shown earlier, the test event uses a simplified format with direct fields. When implementing event handling, ensure your code can distinguish between and properly handle both message formats.

Example message when an object is created using a PUT request

The following is an example of a message that Amazon S3 sends to publish an `s3:ObjectCreated:Put` event.

```
{
   "Records":[
      {
         "eventVersion":"2.6",
         "eventSource":"aws:s3",
         "awsRegion":"us-west-2",
         "eventTime":"1970-01-01T00:00:00.000Z",
         "eventName":"ObjectCreated:Put",
         "userIdentity":{
            "principalId":"AIDAJDPLRKLG7UEXAMPLE"
         },
         "requestParameters":{
            "sourceIPAddress":"172.16.0.1"
         },
         "responseElements":{
            "x-amz-request-id":"C3D13FE58DE4C810",
            "x-amz-id-2":"FMyUVURIY8/IgAtTv8xRjskZQpcIZ9KG4V5Wp6S7S/
JRWeUWerMUE5JgHvANOjpD"
         },
         "s3":{
            "s3SchemaVersion":"1.0",
            "configurationId":"testConfigRule",
            "bucket":{
               "name":"amzn-s3-demo-bucket",
               "ownerIdentity":{
                  "principalId":"A3NL1KOZZKExample"
               },
```

```
                "arn":"arn:aws:s3:::amzn-s3-demo-bucket"
            },
            "object":{
                "key":"HappyFace.jpg",
                "size":1024,
                "eTag":"d41d8cd98f00b204e9800998ecf8427e",
                "versionId":"096fKKXTRTtl3on89fVO.nfljtsv6qko",
                "sequencer":"0055AED6DCD90281E5"
            }
         }
      }
   ]
}
```

Using EventBridge

Amazon S3 can send events to Amazon EventBridge whenever certain events happen in your bucket. Unlike other destinations, you don't need to select which event types you want to deliver. After EventBridge is enabled, all events below are sent to EventBridge. You can use EventBridge rules to route events to additional targets. The following lists the events Amazon S3 sends to EventBridge.

Event type	Description
Object Created	An object was created. The reason field in the event message structure indicates which S3 API was used to create the object: PutObject, POST Object, CopyObject, or CompleteMultipartUpload.
Object Deleted (DeleteObject) *Object Deleted (Lifecycle expiration)*	An object was deleted. When an object is deleted using an S3 API call, the reason field is set to DeleteObject. When an object is deleted by an S3 Lifecycle expiration rule, the reason field is set to Lifecycle Expiration. For more information, see Expiring objects. When an unversioned object is deleted, or a versioned object is permanently deleted, the deletion-type field

Event type	Description
	is set to Permanently Deleted. When a delete marker is created for a versioned object, the `deletion-type` field is set to Delete Marker Created. For more information, see Deleting object versions from a versioning-enabled bucket.
Object Restore Initiated	An object restore was initiated from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive storage class or from S3 Intelligent-Tiering Archive Access or Deep Archive Access tier. For more information, see Working with archived objects.
Object Restore Completed	An object restore was completed.
Object Restore Expired	The temporary copy of an object restored from S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive expired and was deleted.
Object Storage Class Changed	An object was transitioned to a different storage class. For more information, see Transitioning objects using Amazon S3 Lifecycle.
Object Access Tier Changed	An object was transitioned to the S3 Intelligent-Tiering Archive Access tier or Deep Archive Access tier. For more information, see Managing storage costs with Amazon S3 Intelligent-Tiering.
Object ACL Updated	An object's access control list (ACL) was set using `PutObjectAcl`. An event is not generated when a request results in no change to an object's ACL. For more information, see Access control list (ACL) overview.
Object Tags Added	A set of tags was added to an object using `PutObjectTagging`. For more information, see Tagging your objects.

Event type	Description
Object Tags Deleted	All tags were removed from an object using `DeleteObjectTagging`. For more information, see Tagging your objects.
Object Annotation Created	An annotation was created or updated on an object using `PutObjectAnnotation`. For more information, see Annotating your objects.
Object Annotation Removed	An annotation was deleted from an object using `DeleteObjectAnnotation`. For more information, see Annotating your objects.
Object Retention Updated	Amazon S3 sends this event when an object's Object Lock retention settings are changed through a `PutObjectRetention` request. This includes setting fixed or variable retention, releasing an event hold, modifying an event hold duration, and setting or extending a retain-until-date.

Note

For more information about how Amazon S3 event types map to EventBridge event types, see Amazon EventBridge mapping and troubleshooting.

You can use Amazon S3 Event Notifications with EventBridge to write rules that take actions when an event occurs in your bucket. For example, you can have it send you a notification. For more information, see What is EventBridge? in the *Amazon EventBridge User Guide*.

For more information about the actions and data types you can interact with using the EventBridge API, see the Amazon EventBridge API Reference in the *Amazon EventBridge API Reference*.

For information about pricing, see Amazon EventBridge pricing.

Topics

- Amazon EventBridge permissions

- Enabling Amazon EventBridge
- EventBridge event message structure
- Amazon EventBridge mapping and troubleshooting

Amazon EventBridge permissions

Amazon S3 does not require any additional permissions to deliver events to Amazon EventBridge.

Enabling Amazon EventBridge

You can enable Amazon EventBridge by using the S3 console, AWS Command Line Interface (AWS CLI), or Amazon S3 REST API.

> **Note**
>
> After you enable EventBridge, it takes around five minutes for the changes to take effect.

Using the S3 console

To enable EventBridge event delivery in the S3 console.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to enable events for.
4. Choose **Properties**.
5. Navigate to the **Event Notifications** section and find the **Amazon EventBridge** subsection. Choose **Edit**.
6. Under **Send notifications to Amazon EventBridge for all events in this bucket** choose **On**.

Using the AWS CLI

The following example creates a bucket notification configuration for bucket *amzn-s3-demo-bucket1* with Amazon EventBridge enabled.

```
aws s3api put-bucket-notification-configuration --bucket amzn-s3-demo-bucket1 --
notification-configuration='{ "EventBridgeConfiguration": {} }'
```

Using the REST API

You can programmatically enable Amazon EventBridge on a bucket by calling the Amazon S3 REST API. For more information, see PutBucketNotificationConfiguration in the *Amazon Simple Storage Service API Reference*.

The following example shows the XML used to create a bucket notification configuration with Amazon EventBridge enabled.

```
<NotificationConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <EventBridgeConfiguration>
  </EventBridgeConfiguration>
</NotificationConfiguration>
```

Creating EventBridge rules

Once enabled you can create Amazon EventBridge rules for certain tasks. For example, you can send email notifications when an object is created. For a full tutorial, see Tutorial: Send a notification when an Amazon S3 object is created in the *Amazon EventBridge User Guide*.

EventBridge event message structure

The notification message that Amazon S3 sends to publish an event is in the JSON format. When Amazon S3 sends an event to Amazon EventBridge, the following fields are present.

- `version` – Currently 0 (zero) for all events. This is an Amazon EventBridge envelope field and is not used for schema versioning. For event schema versioning, see the *event-version* field in the *detail* field.
- `id` – A UUID generated for every event.
- `detail-type` – The type of event that's being sent. See Using EventBridge for a list of event types.
- `source` – Identifies the service that generated the event.
- `account` – The 12-digit AWS account ID of the bucket owner.
- `time` – The time the event occurred.
- `region` – Identifies the AWS Region of the bucket.

- `resources` – A JSON array that contains the Amazon Resource Name (ARN) of the bucket.
- `detail` – A JSON object that contains information about the event. For more information about what can be included in this field, see Event message detail field.

Event message structure examples

The following are examples of some of the Amazon S3 event notification messages that can be sent to Amazon EventBridge.

Object created

```
{
  "version": "0",
  "id": "17793124-05d4-b198-2fde-7ededc63b103",
  "detail-type": "Object Created",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "event-version": "1.2",
    "bucket": {
      "name": "amzn-s3-demo-bucket1",
      "aws-generated-tags": {
        "aws:cloudformation:stack-id": "arn:aws:cloudformation:us-
east-1:111122223333:stack/amzn-s3-demo-stack/51af3dc0-da77-11e4-872e-1234567db123",
        "aws:cloudformation:logical-id": "S3DemoBucket",
        "aws:cloudformation:stack-name": "amzn-s3-demo-stack"
      }
    },
    "object": {
      "key": "example-key",
      "size": 5,
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "IYV3p45BT0ac8hjHg1houSdS1a.Mro8e",
      "sequencer": "617f08299329d189"
    },
    "request-id": "N4N7GDK58NMKJ12R",
```

```
    "requester": "123456789012",
    "source-ip-address": "192.0.2.1",
    "reason": "PutObject"
  }
}
```

Object deleted (using DeleteObject)

```
{
  "version": "0",
  "id": "2ee9cc15-d022-99ea-1fb8-1b1bac4850f9",
  "detail-type": "Object Deleted",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "event-version": "1.2",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "1QW9g1Z99LUNbvaaYVpW9xDlOLU.qxgF",
      "sequencer": "617f0837b476e463"
    },
    "request-id": "0BH729840619AG5K",
    "requester": "123456789012",
    "source-ip-address": "192.0.2.1",
    "reason": "DeleteObject",
    "deletion-type": "Delete Marker Created"
  }
}
```

Object deleted (using lifecycle expiration)


```
{
  "version": "0",
  "id": "ad1de317-e409-eba2-9552-30113f8d88e3",
  "detail-type": "Object Deleted",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "event-version": "1.2",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "mtB0cV.jejK63XkRNceanNMC.qXPWLeK",
      "sequencer": "617b398000000000"
    },
    "request-id": "20EB74C14654DC47",
    "requester": "s3.amazonaws.com",
    "reason": "Lifecycle Expiration",
    "deletion-type": "Delete Marker Created"
  }
}
```


Object restore completed


```
{
  "version": "0",
  "id": "6924de0d-13e2-6bbf-c0c1-b903b753565e",
  "detail-type": "Object Restore Completed",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2021-11-12T00:00:00Z",
  "region": "ca-central-1",
```

```
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "event-version": "1.2",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "size": 5,
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "KKsjUC1.6gIjqtvhfg5AdMI0eCePIiT3"
    },
    "request-id": "189F19CB7FB1B6A4",
    "requester": "s3.amazonaws.com",
    "restore-expiry-time": "2021-11-13T00:00:00Z",
    "source-storage-class": "GLACIER"
  }
}
```

Object annotation created

```
{
  "version": "0",
  "id": "8c2bfa57-4a67-b2ea-c4e0-6d420ea3b3a6",
  "detail-type": "Object Annotation Created",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2025-03-15T00:00:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "event-version": "1.2",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
```

```
      "key": "example-key",
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "IYV3p45BT0ac8hjHg1houSdS1a.Mro8e",
      "sequencer": "617f08299329d189"
    },
    "request-id": "4B4NGD358NMKJ12R",
    "requester": "123456789012",
    "source-ip-address": "192.0.2.1",
    "reason": "PutObjectAnnotation",
    "object-annotation": {
      "name": "my-annotation",
      "size": 24,
      "etag": "05df351ea190dffc26f55c74425b9d7a"
    }
  }
}
```

Object annotation removed

```
{
  "version": "0",
  "id": "a3d1bc94-5f82-c7e1-d290-7e531fb4c5b8",
  "detail-type": "Object Annotation Removed",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2025-03-15T00:00:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "event-version": "1.2",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "etag": "b1946ac92492d2347c6235b4d2611184",
      "version-id": "IYV3p45BT0ac8hjHg1houSdS1a.Mro8e",
      "sequencer": "617f08299329d190"
    },
```

```
    "request-id": "7C5NGD358NMKJ34S",
    "requester": "123456789012",
    "source-ip-address": "192.0.2.1",
    "reason": "DeleteObjectAnnotation",
    "object-annotation": {
      "name": "my-annotation"
    }
  }
}
```

Object retention updated

```
{
  "version": "0",
  "id": "a1b2c3d4-e5f6-7890-abcd-ef1234567890",
  "detail-type": "Object Retention Updated",
  "source": "aws.s3",
  "account": "111122223333",
  "time": "2026-04-07T14:30:00Z",
  "region": "us-east-1",
  "resources": [
    "arn:aws:s3:::amzn-s3-demo-bucket1"
  ],
  "detail": {
    "version": "0",
    "event-version": "1.2",
    "bucket": {
      "name": "amzn-s3-demo-bucket1"
    },
    "object": {
      "key": "example-key",
      "etag": "d41d8cd98f00b204e9800998ecf8427e",
      "version-id": "abc123",
      "sequencer": "0055AED6DCD90281E5"
    },
    "request-id": "C3D13FE58DE4C810",
    "requester": "111122223333",
    "source-ip-address": "192.0.2.1",
    "object-retention-event-data": {
      "mode": "COMPLIANCE",
      "retain-until-date": "2026-07-06T14:30:00Z",
      "event-hold": "ON",
```

```
      "event-hold-duration": { "days": 90 }
    }
  }
}
```

Event message detail field

The detail field contains a JSON object with information about the event. The following fields may be present in the detail field.

- `version` – Currently 0 (zero) for all events.
- `event-version` – The event schema version in the form *`major`*`.`*`minor`* (for example, `1.1`).

  The major version is incremented if Amazon S3 makes a change to the event structure that's not backward compatible. This includes removing a JSON field that's already present or changing how the contents of a field are represented (for example, a date format).

  The minor version is incremented if Amazon S3 makes a backward-compatible change to the event structure. This includes adding new fields to the event structure or introducing new event types. To stay compatible with new minor versions of the event structure, we recommend that your applications ignore new fields.

  To ensure that your applications can parse the event structure correctly, we recommend that you do an equal-to comparison on the major version number. To ensure that the fields that are expected by your application are present, we also recommend doing a greater-than-or-equal-to comparison on the minor version.
- `bucket` – Information about the Amazon S3 bucket involved in the event.
- `aws-generated-tags` – Tags attached to the bucket by AWS services. Only present when AWS generated tags exist on the bucket. This field is not present for cross-Region replication events.
- `object` – Information about the Amazon S3 object involved in the event.
- `request-id` – Request ID in S3 response.
- `requester` – AWS account ID or AWS service principal of requester.
- `source-ip-address` – Source IP address of S3 request. Only present for events triggered by an S3 request.
- `reason` – For **Object Created** events, the S3 API used to create the object: PutObject, POST Object, CopyObject, or CompleteMultipartUpload. For **Object Deleted** events, this is set to

DeleteObject when an object is deleted by an S3 API call, or **Lifecycle Expiration** when an object is deleted by an S3 Lifecycle expiration rule. For more information, see Expiring objects.

- `deletion-type` – For **Object Deleted** events, when an unversioned object is deleted, or a versioned object is permanently deleted, this is set to **Permanently Deleted**. When a delete marker is created for a versioned object, this is set to **Delete Marker Created**. For more information, see Deleting object versions from a versioning-enabled bucket.

> **Note**
>
> Some object attributes (such as `etag` and `size`) are present only when a delete marker is created.

- `restore-expiry-time` – For **Object Restore Completed** events, the time when the temporary copy of the object will be deleted from S3. For more information, see Working with archived objects.
- `source-storage-class` – For **Object Restore Initiated** and **Object Restore Completed** events, the storage class of the object being restored. For more information, see Working with archived objects.
- `destination-storage-class` – For **Object Storage Class Changed** events, the new storage class of the object. For more information, see Transitioning objects using Amazon S3 Lifecycle.
- `destination-access-tier` – For **Object Access Tier Changed** events, the new access tier of the object. For more information, see Managing storage costs with Amazon S3 Intelligent-Tiering.
- `object-annotation` – For **Object Annotation Created** and **Object Annotation Removed** events, the `detail` field includes an `object-annotation` field containing information about the annotation.
- `has-object-annotation` – For **Object Created** events with reason `CopyObject`, the `object` block includes a `has-object-annotation` boolean field that indicates whether the copied object has annotations.
- `object-retention-event-data` – For **Object Retention Updated** events, a block that contains the object's Object Lock retention state after the change. This block contains the following fields.
  - `mode` – The Object Lock retention mode applied to the object after the change: `COMPLIANCE` or `GOVERNANCE`.

- `retain-until-date` – The date in ISO-8601 format until which the object is retained after the change. For objects with an active event hold, this is the dynamically computed date at the time Amazon S3 processes the event.
- `event-hold` – The event hold state after the change: `ON` or `OFF`. Only present for event-hold operations (setting or releasing an event hold).
- `event-hold-duration` – The event hold duration as `{"days": N}` or `{"years": N}`. Only present when an event hold duration is set or modified.

Amazon EventBridge mapping and troubleshooting

The following table describes how Amazon S3 event types are mapped to Amazon EventBridge event types.

S3 event type	Amazon EventBridge detail type
ObjectCreated:Put ObjectCreated:Post ObjectCreated:Copy ObjectCreated:CompleteMultipartUpload	Object Created
ObjectRemoved:Delete ObjectRemoved:DeleteMarkerCreated LifecycleExpiration:Delete LifecycleExpiration:DeleteMarkerCreated	Object Deleted
ObjectRestore:Post	Object Restore Initiated
ObjectRestore:Completed	Object Restore Completed
ObjectRestore:Delete	Object Restore Expired

S3 event type	Amazon EventBridge detail type
LifecycleTransition	Object Storage Class Changed
IntelligentTiering	Object Access Tier Changed
ObjectTagging:Put	Object Tags Added
ObjectTagging:Delete	Object Tags Deleted
ObjectAcl:Put	Object ACL Updated
ObjectAnnotation:Put	Object Annotation Created
ObjectAnnotation:Delete	Object Annotation Removed
ObjectRetention:Put	Object Retention Updated

Amazon EventBridge troubleshooting

For information about how to troubleshoot EventBridge, see Troubleshooting Amazon EventBridge in the *Amazon EventBridge User Guide*.

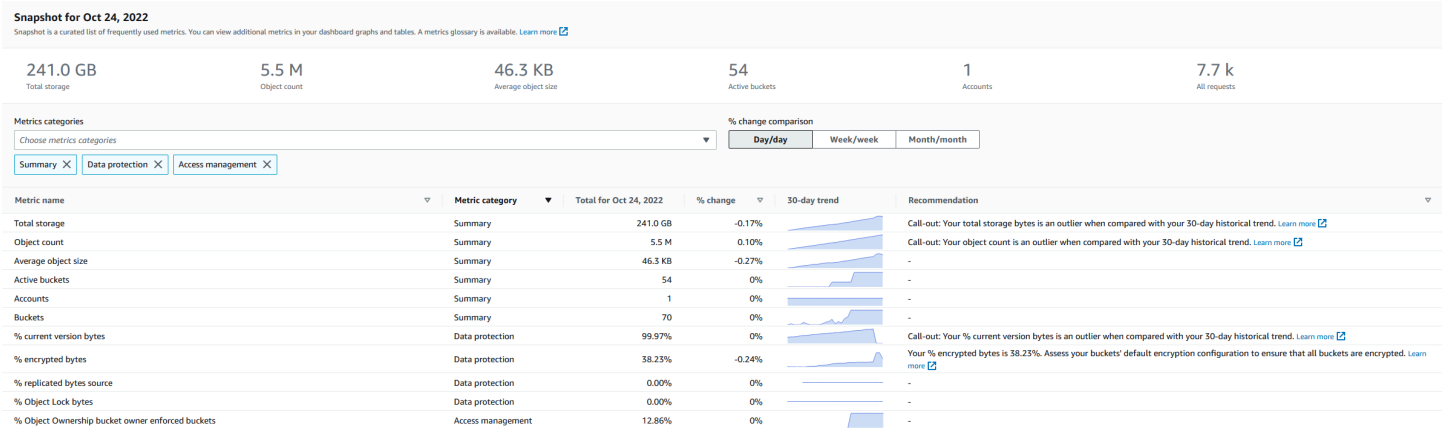
Monitoring your storage activity and usage with Amazon S3 Storage Lens

Amazon S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object storage and activity. S3 Storage Lens also analyzes metrics to deliver contextual recommendations that you can use to optimize storage costs and apply best practices for protecting your data.

You can use S3 Storage Lens metrics to generate summary insights. For example, you can find out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use S3 Storage Lens metrics to identify cost optimization opportunities, implement data protection and access management best practices, and improve the performance of application workloads. For example, you can identify buckets that don't have S3 Lifecycle rules set up to expire incomplete multipart uploads that are more than 7 days old.

You can also identify buckets that aren't following data protection best practices, such as using S3 Replication or S3 Versioning.

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics report in CSV or Parquet format to a general purpose S3 bucket or export the metrics directly to an AWS-managed S3 table bucket.



S3 Storage Lens metrics and features

S3 Storage Lens provides an interactive *default dashboard* that is updated daily. S3 Storage Lens preconfigures this dashboard to visualize the summarized insights and trends for your entire account and updates them daily in the S3 console. Metrics from this dashboard are also summarized in your account snapshot on the **Buckets** page. For more information, see Default dashboard.

To create other dashboards and scope them by AWS Regions, S3 buckets, or accounts (for AWS Organizations), you create an S3 Storage Lens dashboard configuration. You can create and manage S3 Storage Lens dashboard configurations by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API. When you create or edit an S3 Storage Lens dashboard, you define your dashboard scope and metrics selection.

S3 Storage Lens offers free tier metrics and advanced tier metrics, which you can upgrade to for an additional charge. With the advanced tier, you can access additional metrics and features for gaining insight into your storage. These features include advanced metric categories, prefix

aggregation, contextual recommendations, expanded prefixes metrics reports, and Amazon CloudWatch publishing. Prefix aggregation and contextual recommendations are available only in the Amazon S3 console. For information about S3 Storage Lens pricing, see Amazon S3 pricing.

Metrics categories

Within the free and advanced tiers, metrics are organized into categories that align with key use cases, such as cost optimization and data protection. Free metrics include summary, cost optimization, data protection, access management, performance, and event metrics. When you upgrade to the advanced tier, you can enable advanced cost optimization and data protection metrics. You can use these advanced metrics to further reduce your S3 storage costs and improve your data protection stance. You can also enable activity metrics and detailed status-code metrics to improve the performance of application workloads that are accessing your S3 buckets. For more information about the free and advanced metrics categories, see Metrics selection.

You can assess your storage based on S3 best practices, such as analyzing the percentage of your buckets that have encryption or S3 Object Lock or S3 Versioning enabled. You can also identify potential cost-savings opportunities. For example, you can use S3 Lifecycle rule count metrics to identify buckets that are missing lifecycle expiration or transition rules. You can also analyze your request activity per bucket to find buckets where objects could be transitioned to a lower-cost storage class. For more information, see Amazon S3 Storage Lens metrics use cases.

Metrics export

Default metrics report

The default metrics report in S3 Storage Lens includes free metrics and advanced tier metrics covering object storage usage and activity trends across your AWS accounts. The report includes prefix aggregation for prefixes whose objects comprise at least 1% of the total data stored in the bucket, and supports up to 10 levels of prefix depth. The report can be exported daily in CSV or Parquet format to an S3 general purpose bucket. The report can also be sent to an AWS-managed S3 table bucket (with name `aws-s3`) making it easy to query using AWS analytics services or third-party tools.

With the default metrics report, you can identify cost optimization opportunities like buckets without S3 Lifecycle rules for incomplete multipart uploads and buckets not following data protection best practices such as S3 Replication or S3 Versioning. The default metrics report also provides contextual recommendations for optimizing storage costs and applying data protection best practices, at no additional charge beyond standard S3 storage costs.

Expanded prefixes metrics report

The Storage Lens expanded prefixes metrics report provides comprehensive prefix-level analytics across your entire S3 storage data, expanding coverage to support billions of prefixes in your bucket. This report delivers metrics for all prefixes in your buckets, including storage usage, bytes transferred, request counts by status code, and data protection compliance metrics, which you can export daily in CSV or Parquet format to S3 general purpose bucket. You can also export the metrics directly to the `aws-s3` AWS-managed S3 table bucket.

> **Note**
>
> The report processes metrics for prefixes up to 50 levels deep and excludes prefix-level metrics for any bucket where the prefix and storage class combinations exceed twice the object count.

With the expanded prefixes metrics report, you can identify performance optimization opportunities, such as high error rates, small objects, or sub-optimal request patterns, across billions of prefixes in your bucket. Unlike the default metrics report, the expanded prefixes metrics report delivers metrics for granular prefixes in your bucket. For example, you can identify prefixes with large numbers of objects of size less than 128KB to quickly isolate such datasets for compaction that will improve application performance. This report is available in all AWS Regions as an opt-in feature in the Storage Lens advanced tier dashboard configuration.

Metrics publishing

Amazon CloudWatch publishing

You can publish S3 Storage Lens usage and activity metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch features, such as alarms and triggered actions, metric math, and anomaly detection, to monitor and take action on S3 Storage Lens metrics. In addition, CloudWatch API operations enable applications, including third-party providers, to access your S3 Storage Lens metrics. The CloudWatch publishing option is available for dashboards that are upgraded to the S3 Storage Lens advanced tier. For more information about support for S3 Storage Lens metrics in CloudWatch, see Monitor S3 Storage Lens metrics in CloudWatch.

For more information about using S3 Storage Lens, see the following topics.

Topics

- Understanding Amazon S3 Storage Lens
- Amazon S3 Storage Lens metrics glossary
- Setting Amazon S3 Storage Lens permissions
- Working with Amazon S3 Storage Lens by using the console and API
- Viewing metrics with Amazon S3 Storage Lens
- Viewing metrics in AWS managed S3 Tables
- Using Amazon S3 Storage Lens with AWS Organizations
- Working with S3 Storage Lens groups to filter and aggregate metrics

Understanding Amazon S3 Storage Lens

> **Important**
>
> Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

Amazon S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. You can use S3 Storage Lens metrics to generate summary insights, such as finding out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use S3 Storage Lens metrics to identify cost-optimization opportunities, implement data-protection and security best practices, and improve the performance of application workloads. For example, you can identify buckets that don't have S3 Lifecycle rules to expire incomplete multipart uploads that are more than 7 days old. You can also identify buckets that aren't following data-protection best practices, such as using S3 Replication or S3 Versioning. S3 Storage Lens also analyzes metrics to deliver contextual recommendations that you can use to optimize storage costs and apply best practices for protecting your data.

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics report in CSV or Parquet format to a general purpose S3 bucket or export the metrics directly to an AWS-managed S3 table bucket. You can create and manage S3 Storage Lens dashboards by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or Amazon S3 REST API.

S3 Storage Lens concepts and terminology

This section contains the terminology and concepts that are essential for successfully understanding and using Amazon S3 Storage Lens.

Topics

- Dashboard configuration
- Default dashboard
- Dashboards
- Account snapshot
- Metrics export
- Metrics export destinations
- Home Region
- Retention period
- Metrics categories
- Recommendations
- Metrics selection
- Prefix delimiter
- S3 Storage Lens and AWS Organizations

Dashboard configuration

S3 Storage Lens requires a dashboard configuration that contains the properties required to aggregate metrics on your behalf for a single dashboard or export. When you create a

configuration, you choose the dashboard name and the home Region, which you can't change after you create the dashboard. You can optionally add tags and configure a metrics export in CSV or Parquet format.

In the dashboard configuration, you also define the dashboard scope and the metrics selection. The scope can include all the storage for your organization account or sections that are filtered by Region, bucket, and account. When you configure the metrics selection, you choose between free tier metrics and advanced tier metrics, which you can upgrade to for an additional charge. With the advanced tier, you can access additional metrics and features. These features include advanced metric categories, prefix-level aggregation, contextual recommendations, and Amazon CloudWatch publishing. For information about S3 Storage Lens pricing, see Amazon S3 pricing.

Default dashboard

The S3 Storage Lens default dashboard on the console is named **default-account-dashboard**. S3 preconfigures this dashboard to visualize the summarized insights and trends for your entire account and updates them daily in the S3 console. You can't modify the configuration scope of the default dashboard, but you can upgrade the metrics selection from free tier metrics to advanced tier metrics. You can configure the optional metrics export or even disable the dashboard. However, you can't delete the default dashboard.

> **Note**
>
> If you disable your default dashboard, it's no longer updated. You'll no longer receive any new daily metrics in your S3 Storage Lens dashboard, your metrics export, or the account snapshot on the S3 **Buckets** page. If your dashboard uses advanced metrics, you'll no longer be charged. You can still see historic data in the dashboard until the 14-day period for data queries expires. This period is 15 months if you've enabled advanced metrics. To access historic data, you can re-enable the dashboard within the expiration period.

Dashboards

You can create additional S3 Storage Lens dashboards and scope them by AWS Regions, S3 buckets, or accounts (for AWS Organizations). When you create or edit a S3 Storage Lens dashboard, you define your dashboard scope and metrics selection. S3 Storage Lens offers free tier metrics and advanced tier metrics, which you can upgrade to for an additional charge. With advanced metrics, you can access additional metrics and features for gaining insight into

your storage. These include advanced metric categories, prefix-level aggregation, contextual recommendations, and Amazon CloudWatch publishing. For information about S3 Storage Lens pricing, see Amazon S3 pricing.

You can also disable or delete dashboards. If you disable a dashboard, it's no longer updated, and you will no longer receive any new daily metrics. You can still see historic data until the 14-day expiration period. If you enabled advanced metrics for that dashboard, this period is 15 months. To access historic data, you can re-enable the dashboard within the expiration period.

If you delete your dashboard, you lose all your dashboard configuration settings. You will no longer receive any new daily metrics, and you also lose access to the historical data associated with that dashboard. If you want to access the historic data for a deleted dashboard, you must create another dashboard with the same name in the same home Region.

Note

- You can use S3 Storage Lens to create up to 50 dashboards per home Region.
- Organization-level dashboards can be limited only to a Regional scope.

Account snapshot

The S3 Storage Lens **Account snapshot** summarizes metrics from your default dashboard and displays your total storage, object count, and average object size on the S3 console **Buckets** page. This account snapshot gives you quick access to insights about your storage without having to leave the **Buckets** page. The account snapshot also provides one-click access to your interactive S3 Storage Lens dashboard.

You can use your dashboard to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data protection best practices. Your dashboard has drill-down options to generate insights at the organization, account, bucket, object, or prefix level. You can also send a once-daily metrics export to an S3 bucket in CSV or Parquet format.

You can't modify the dashboard scope of the **default-account dashboard** because it's linked to the **Account snapshot**. However, you can upgrade the metrics selection in your **default-account-dashboard** from free metrics to paid advanced metrics. After upgrading, you can then display all requests, bytes uploaded, and bytes downloaded in the S3 Storage Lens **Account snapshot**.

> **Note**
>
> If you disable your default dashboard, your **Account snapshot** is no longer updated. To continue displaying metrics in the **Account snapshot**, you can re-enable the **default-account-dashboard**.

Metrics export

An S3 Storage Lens metrics export is a file that contains all the metrics identified in your S3 Storage Lens configuration. This information is generated daily in CSV or Parquet format and is sent to a general purpose S3 bucket. You can also export the metrics directly to the `aws-s3` AWS-managed S3 table bucket making it easy to query using AWS analytics services or third-party tools. You can use the metrics export for further analysis by using the metrics tool of your choice. The bucket specified for your metrics export must be in the same Region as your S3 Storage Lens configuration. You can generate an S3 Storage Lens metrics export from the S3 console by editing your dashboard configuration. You can also configure a metrics export by using the AWS CLI and AWS SDKs.

There are two types of metric exports available in Storage Lens:

- **Default metrics report** – The default metrics report in S3 Storage Lens includes free metrics and activity trends across your AWS account and aggregates usage metrics for top prefixes.
- **Expanded prefixes metrics report** – The Storage Lens expanded prefixes metrics report provides granular storage and activity metrics (such as storage usage, bytes transferred, and request counts by status code) at the prefix level for every prefix in your bucket. This report is available as an opt-in feature in all AWS Regions, through the advanced pricing tier in your Storage Lens dashboard configuration. For information about S3 Storage Lens feature pricing, see Amazon S3 pricing.

> **Note**
>
> Storage Lens only generates metrics for S3 general purpose buckets.

Metrics export destinations

When exporting Storage Lens metrics data, you can choose both an S3 general purpose bucket or an S3 table bucket as your destination. General purpose buckets provide broad compatibility with existing tools and applications, offering flexibility to process data within your account, using your preferred analytics services. This option supports standard S3 access patterns and integrations for data analysis within individual buckets in your Region. In contrast, S3 table bucket lets you run immediate queries across multiple accounts and regions, create custom dashboards with AWS Quick, and join data with other AWS services or third-party tools, without the need for additional processing infrastructure. For example, you can combine Storage Lens metrics with S3 Metadata to analyze object activity patterns across your organization.

S3 general purpose bucket

Exporting Storage Lens metrics to an S3 general purpose bucket offers flexibility and continuity for storing your Storage Lens data. You can maintain existing workflows and operational consistency by continuing to use your current infrastructure and existing extract, transform, and load (ETL) processes, analytics tools, or automated workflows. General purpose buckets also work with the full range of AWS services and third-party tools that support standard S3 APIs. This gives you maximum flexibility in how you process, analyze, or visualize your Storage Lens insights. Additionally, you can implement S3 lifecycle policies to automatically manage data retention, transitioning older metrics to lower-cost storage classes or deleting them after specified periods to optimize costs. Therefore, if operational continuity and workflow flexibility are your priorities for Storage Lens implementation, then consider choosing an S3 general purpose bucket for exporting your Storage Lens data. For more information about S3 general purpose buckets pricing, see Amazon S3 pricing.

S3 table bucket

When exporting Storage Lens metrics to S3 table bucket, you can easily analyze your storage usage and activity metrics without building data pipelines. Your metrics are organized in S3 Tables that are created in an AWS-managed S3 table bucket called `aws-s3` for optimal query performance, with customizable retention periods and encryption settings to meet your data management needs. With your metrics in S3 Tables, you can run queries across multiple accounts and Regions using SQL tools and AWS analytics services (like Amazon Athena, AWS Quick, Amazon EMR, and Amazon Redshift) to create custom dashboards and generate deeper insights. For example, you can join S3 Storage Lens metrics with S3 Metadata to identify objects in prefixes that aren't showing any recent activity. Any data stored in an S3 table bucket incurs S3 Tables costs. For more information about S3 Tables pricing, see Amazon S3 pricing.

Home Region

The home Region is the AWS Region where all S3 Storage Lens metrics for a given dashboard configuration are stored. You must choose a home Region when you create your S3 Storage Lens dashboard configuration. After you choose a home Region, you can't change it. Also, if you're creating a Storage Lens group, we recommend that you choose the same home Region as your Storage Lens dashboard.

Note

You can choose one of the following Regions as your home Region:

- US East (N. Virginia) – `us-east-1`
- US East (Ohio) – `us-east-2`
- US West (N. California) – `us-west-1`
- US West (Oregon) – `us-west-2`
- Asia Pacific (Mumbai) – `ap-south-1`
- Asia Pacific (Seoul) – `ap-northeast-2`
- Asia Pacific (Singapore) – `ap-southeast-1`
- Asia Pacific (Sydney) – `ap-southeast-2`
- Asia Pacific (Tokyo) – `ap-northeast-1`
- Canada (Central) – `ca-central-1`
- China (Beijing) – `cn-north-1`
- China (Ningxia) – `cn-northwest-1`
- Europe (Frankfurt) – `eu-central-1`
- Europe (Ireland) – `eu-west-1`
- Europe (London) – `eu-west-2`
- Europe (Paris) – `eu-west-3`
- Europe (Stockholm) – `eu-north-1`
- South America (São Paulo) – `sa-east-1`

Retention period

S3 Storage Lens metrics are retained so that you can see historical trends and compare differences in your storage and activity over time. You can use Amazon S3 Storage Lens metrics for queries so that you can see historical trends and compare differences in your storage usage and activity over time.

All S3 Storage Lens metrics are retained for a period of 15 months. However, metrics are only available for queries for a specific duration, which depends on your metrics selection. This duration can't be modified. Free metrics are available for queries for a 14-day period, and advanced metrics are available for queries for a 15-month period.

Metrics categories

Within the free and advanced tiers, S3 Storage Lens metrics are organized into categories that align with key use cases, such as cost optimization and data protection. Free metrics include summary, cost optimization, data protection, access management, performance, and event metrics. When you upgrade to advanced metrics, you can enable additional cost optimization and data protection metrics that you can use to further reduce your S3 storage costs and ensure your data is protected. You can also enable activity metrics and detailed status-code metrics that you can use to improve the performance of application workflows.

The following list shows all of the free and advanced metric categories. For a complete list of the individual metrics included in each category, see Amazon S3 Storage Lens metrics glossary.

Summary metrics

Summary metrics provide general insights about your S3 storage, including your total storage bytes and object count.

Cost optimization metrics

Cost optimization metrics provide insights that you can use to manage and optimize your storage costs. For example, you can identify buckets that have incomplete multipart uploads that are more than 7-days old.

With advanced metrics, you can enable advanced cost optimization metrics. These metrics include S3 Lifecycle rule count metrics that you can use to get per-bucket expiration and transition S3 Lifecycle rule counts.

Data-protection metrics

Data-protection metrics provide insights for data protection features, such as encryption and S3 Versioning. You can use these metrics to identify buckets that are not following data protection best practices. For example, you can identify buckets that are not using default encryption with AWS Key Management Service keys (SSE-KMS) or S3 Versioning.

With advanced metrics, you can enable advanced data protection metrics. These metrics include per-bucket replication rule count metrics.

Access management metrics

Access management metrics provide insights for S3 Object Ownership. You can use these metrics to see which Object Ownership settings your buckets use.

Event metrics

Event metrics provide insights for S3 Event Notifications. With event metrics, you can see which buckets have S3 Event Notifications configured.

Performance metrics

Performance metrics provide insights for S3 Transfer Acceleration. With performance metrics, you can see which buckets have Transfer Acceleration enabled.

Activity metrics (advanced)

If you upgrade your dashboard to the **Advanced tier**, you can enable activity metrics. Activity metrics provide details about how your storage is requested (for example, all requests, Get requests, Put requests), bytes uploaded or downloaded, and errors.

Prefix-level activity metrics can be used to help you determine which prefixes are being used infrequently, so that you can transition to a more optimal storage class using S3 Lifecycle.

Detailed status code metrics (advanced)

If you upgrade your dashboard to the **Advanced tier**, you can enable detailed status code metrics. Detailed status code metrics provide insights for HTTP status codes, such as 403 Forbidden and 503 Service Unavailable, that you can use to troubleshoot access or performance issues. For example, you can look at the **403 Forbidden error count** metric to identify workloads that are accessing buckets without the correct permissions applied.

Prefix-level detailed status code metrics can be used to gain a better understanding of the HTTP status code occurrences by prefix. For example, 503 error count metrics enable you to identify prefixes receiving throttling requests during data ingestion.

Advanced cost optimization metrics

Advanced cost optimization metrics provide detailed insights into your S3 lifecycle management configurations to help you optimize storage costs through automated data transitions and deletions. These metrics track the number of lifecycle rules configured across different lifecycle rule types. You can use these metrics to ensure comprehensive lifecycle rule coverage across your buckets and identify opportunities to implement additional cost optimization strategies through automated data management.

Advanced data protection metrics

Advanced data protection metrics help you protect your data by providing insights into replication rule counts, SSE-KMS encryption usage, and security vulnerabilities such as unsupported signature and TLS requests. (**Note:** Replication rule count metrics aren't available for prefixes.)

This visibility enables you to ensure proper data redundancy, validate encryption compliance, identify security risks from outdated protocols, troubleshoot replication misconfigurations, and maintain robust data protection strategies at the organization, account, and bucket levels.

Advanced performance metrics

Advanced performance metrics reveal how your applications interact with data in S3 and can help identify opportunities to optimize application performance such as inefficient I/O patterns, cross-region access, and unique object access count. Storage Lens advanced performance metrics eliminates the need for expensive custom monitoring tools and enables customers to implement S3 best practices more effectively, particularly benefiting performance sensitive applications such as machine learning training, data analytics, and other high-performance compute workloads.

Recommendations

S3 Storage Lens provides automated recommendations to help you optimize your storage. Recommendations are placed contextually alongside relevant metrics in the S3 Storage Lens dashboard. Historical data is not eligible for recommendations because recommendations are relevant to what is happening in the most recent period. Recommendations appear only when they are relevant.

S3 Storage Lens recommendations come in the following forms:

- **Suggestions**

Suggestions alert you to trends within your storage and activity that might indicate a storage-cost optimization opportunity or a data protection best practice. You can use the suggested topics in the *Amazon S3 User Guide* and the S3 Storage Lens dashboard to drill down for more details about the specific Regions, buckets, or prefixes.

- **Call-outs**

  Call-outs are recommendations that alert you to interesting anomalies within your storage and activity over a period that might need further attention or monitoring.

  - **Outlier call-outs**

    S3 Storage Lens provides call-outs for metrics that are outliers, based on your recent 30-day trend. The outlier is calculated by using a standard score, also known as a *z-score*. In this score, the current day's metric is subtracted from the average of the last 30 days for that metric. The current day's metric is then divided by the standard deviation for that metric over the last 30 days. The resulting score is usually between -3 and +3. This number represents the number of standard deviations that the current day's metric is from the mean.

    S3 Storage Lens considers metrics with a score >2 or <-2 to be outliers because they are higher or lower than 95 percent of normally distributed data.

  - **Significant change call-outs**

    The significant change call-out applies to metrics that are expected to change less frequently. Therefore, it's set to a higher sensitivity than the outlier calculation, which is typically in the range of +/- 20 percent versus the prior day, week, or month.

    Addressing call-outs in your storage and activity – If you receive a significant change call-out, it’s not necessarily a problem. The call-out could be the result of an anticipated change in your storage. For example, you might have recently added a large number of new objects, deleted a large number of objects, or made similar planned changes.

    If you see a significant change call-out on your dashboard, take note of it and determine whether it can be explained by recent circumstances. If not, use the S3 Storage Lens dashboard to drill down for more details to understand the specific Regions, buckets, or prefixes that are driving the fluctuation.

- **Reminders**

Reminders provide insights into how Amazon S3 works. They can help you learn more about ways to use S3 features to reduce storage costs or apply data protection best practices.

Metrics selection

S3 Storage Lens offers two metrics selections that you can choose for your dashboard and export: *free tier* and *advanced tier*.

- **Free tier**

  S3 Storage Lens offers free metrics for all dashboards and configurations. Free metrics contain metrics that are relevant to your storage, such as the number of buckets and the objects in your account. Free metrics also include use-case based metrics (for example, cost optimization and data protection metrics) that you can use to investigate whether your storage is configured according to S3 best practices. All free tier metrics are collected daily and can be exported to either an S3 general purpose bucket (CSV or Parquet format) or S3 table bucket (Parquet format only). Data is available for queries for 14 days in the Amazon S3 console. For more information about which metrics are available with free metrics, see the Amazon S3 Storage Lens metrics glossary.

- **Advanced tier**

  S3 Storage Lens offers free metrics for all dashboards and configurations with the option to upgrade to advanced metrics. Additional charges apply. For more information, see Amazon S3 pricing.

  Advanced tier metrics include all the metrics in free metrics along with additional metrics, such as advanced data protection and cost optimization metrics, activity metrics, and detailed status-code metrics. Advanced tier metrics also provide recommendations to help you optimize your storage. Recommendations are placed contextually alongside relevant metrics in the dashboard.

  Advanced tier includes the following features:

  - **Advanced metrics categories** – Generate additional metrics. For a complete list of advanced metric categories, see Metrics categories. For a complete list of metrics, see the Amazon S3 Storage Lens metrics glossary.
  - **Amazon CloudWatch publishing** – Publishes S3 Storage Lens metrics to CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch API operations and features, such as alarms and triggered actions, metric math,

and anomaly detection, to monitor and take action on S3 Storage Lens metrics. For more information, see Monitor S3 Storage Lens metrics in CloudWatch.

- **Default metrics report** – The default metrics report in S3 Storage Lens includes free metrics and prefix aggregation capabilities for top prefixes for object storage usage and activity trends across your AWS accounts. With the default metrics report, you can identify cost optimization opportunities at no additional charge beyond standard S3 storage costs.
- **Expanded prefixes metrics report** – The Storage Lens expanded prefixes metrics report provides comprehensive prefix-level analytics across your entire S3 storage data, expanding coverage to support up to billions of prefixes per bucket.
- **Additional metrics aggregation**
  - **Prefix aggregation** – Collects metrics at the prefix level. This setting specifies the prefixes aggregated as part of the default metrics report, which is displayed in the Storage Lens dashboard. Note that metrics that are applicable at the prefix level are available with **Prefix aggregation**, except for bucket-level settings and rule count metrics. Prefix-level metrics don't apply to the expanded prefixes metrics export and aren't published to CloudWatch.
  - **Storage Lens group aggregation** – Collects metrics at the Storage Lens group level. After you enable the advanced tier metrics and Storage Lens group aggregation, you can specify which Storage Lens groups to include or exclude from your Storage Lens dashboard. At least one Storage Lens group must be specified. Storage Lens groups that are specified must also reside within the designated home Region in the dashboard account. Storage Lens group-level metrics are not published to CloudWatch.

All advanced metrics are collected daily. Data is available for querying for up to 15 months in the Amazon S3 console. For more information about the storage metrics that are aggregated by S3 Storage Lens, see Amazon S3 Storage Lens metrics glossary.

Prefix delimiter

Prefix delimiters determine how Storage Lens counts prefix depth, by separating the hierarchical levels within object keys. You can only specify a single character to indicate each level within your prefixes. If the prefix delimiter is undefined, Amazon S3 uses "/" as the default delimiter.

> **Note**
>
> When you're updating your Storage Lens dashboard configuration through the API, the *delimiter* and the updated *prefix delimiter* must be defined in the same way, or you'll

receive an error. The delimiter only applies to prefix-level metrics that are exported to the default metrics report. The prefix delimiter applies to all prefixes that are exported to the expanded prefixes metrics report.

S3 Storage Lens and AWS Organizations

AWS Organizations is an AWS service that helps you aggregate all of your AWS accounts under one organization hierarchy. Amazon S3 Storage Lens works with AWS Organizations to provide a single view of object storage and activity across your Amazon S3 storage.

For more information, see Using Amazon S3 Storage Lens with AWS Organizations.

- **Trusted access**

  Using your organization's management account, you must enable trusted access for S3 Storage Lens to aggregate storage metrics and usage data for all member accounts in your organization. You can then create dashboards or exports for your organization by using your management account or by giving delegated administrator access to other accounts in your organization.

  You can disable trusted access for S3 Storage Lens at any time, which stops S3 Storage Lens from aggregating metrics for your organization.

- **Delegated administrator**

  You can create dashboards and metrics for S3 Storage Lens for your organization by using your AWS Organizations management account, or by giving *delegated administrator* access to other accounts in your organization. You can deregister delegated administrators at any time. Deregistering a delegated administrator also automatically stops all organization-level dashboards created by that delegated administrator from aggregating new storage metrics.

For more information, see Amazon S3 Storage Lens and AWS Organizations in the *AWS Organizations User Guide*.

Amazon S3 Storage Lens service-linked roles

Along with AWS Organizations trusted access, Amazon S3 Storage Lens uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that's linked directly to S3 Storage Lens. Service-linked roles are predefined by S3 Storage Lens

and include all the permissions that it requires to collect daily storage and activity metrics from member accounts in your organization.

For more information, see Using service-linked roles for Amazon S3 Storage Lens.

Amazon S3 Storage Lens metrics glossary

The Amazon S3 Storage Lens metrics glossary provides a complete list of free and advanced metrics for S3 Storage Lens.

S3 Storage Lens offers free metrics for all dashboards and configurations, with the option to upgrade to advanced metrics.

- **Free metrics** contain metrics that are relevant to your storage usage, such as the number of buckets and the objects in your account. Free metrics also include use-case based metrics, such as cost-optimization and data-protection metrics. All free metrics are collected daily, and data is available for queries for up to 14 days.
- **Advanced metrics** include all the metrics in free metrics along with additional metrics, such as advanced performance, advanced data protection, and advanced cost optimization metrics. Advanced metrics also include additional metric categories, such as activity metrics and detailed status-code metrics. Advanced metrics data is available for queries for 15 months.

  There are additional charges when you use S3 Storage Lens with advanced metrics and recommendations. For more information, see Amazon S3 pricing. For more information about advanced metrics and recommendations features, see Metrics selection.

> **Note**
>
> For Storage Lens groups, only free tier storage metrics are available. Advanced tier metrics are not available at the Storage Lens group level.

Metric names

The **Metric name** column in the following table provides the name of each S3 Storage Lens in the S3 console. The **CloudWatch and export** column provides the name of each metric in Amazon CloudWatch and the metrics export file that you can configure in your S3 Storage Lens dashboard.

Derived metric formulas

Derived metrics are not available for the metrics export and the CloudWatch publishing option. However, you can use the metrics formulas shown in the **Derived metrics formula** column to compute them.

Interpreting the Amazon S3 Storage Lens prefix symbols for metrics unit multiples (K, M, G, and so on)

S3 Storage Lens metrics unit multiples are written with prefix symbols. These prefix symbols match the International System of Units (SI) symbols that are standardized by the International Bureau of Weights and Measures (BIPM). These symbols are also used in the Unified Code for Units of Measure (UCUM). For more information, see List of SI prefix symbols.

Note

- The unit of measurement for S3 storage bytes is in binary gigabytes (GB), where 1 GB is 2^{30} bytes, 1 TB is 2^{40} bytes, and 1 PB is 2^{50} bytes. This unit of measurement is also known as a gibibyte (GiB), as defined by the International Electrotechnical Commission (IEC).
- When an object reaches the end of its lifetime based on its lifecycle configuration, Amazon S3 queues the object for removal and removes it asynchronously. Therefore, there might be a delay between the expiration date and the date when Amazon S3 removes an object. S3 Storage Lens doesn't include metrics for objects that have expired but haven't been removed. For more information about expiration actions in S3 Lifecycle, see Expiring objects.
- Amazon S3 stores metadata (object key, timestamps, etc.) for every object, which requires minimum storage even for 0KB data files. This is why 0KB objects appear in the (0KB-128KB] size range in S3 Storage Lens.
- S3 Storage Lens provides best-effort tracking of cross-region data transfers, primarily focusing on requests from customer-managed resources like EC2 instances. Requests made through AWS PrivateLink or certain in-Region requests are unclassified.

The following table shows the S3 Storage Lens metrics glossary.

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met forn	Storage Lens groups
Total storage	StorageBytes	Total storage, inclusive of incomplet e multipart uploads, object metadata, and delete markers	Free	Sum	N	-	Y
Object count	ObjectCount	Total number of objects inclusive of current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Free	Sum	N	-	Y
Average object size	-	Average object size	Free	Sum	Y	sum geBy sum tCou	Y
Active buckets	-	Number of buckets with storage > 0 bytes	Free	Sum	Y	-	Y
Buckets	-	Number of buckets	Free	Sum	Y	-	Y
Accounts	-	Number of accounts whose	Free	Sum	Y	-	Y

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
		storage is in scope					
Current version bytes	CurrentVersionStorageBytes	Number of bytes that are a current version of an object	Free	Cost optimization	N	-	Y
% current version bytes	-	Percentage of bytes in scope that are current versions of objects	Free	Cost optimization	Y	sum ntVe Stor tes)/ sum Stor tes)	Y
Current version object count	CurrentVersionObjectCount	Count of current version objects, inclusive of current delete markers	Free	Data prot n	N	-	Y
% current version objects	-	Percentage of objects in scope that are a current version	Free	Cost optimization	Y	sum ntVe Obje nt)/ sum bjec t)	Y
Noncurrent version bytes	NonCurrentVersionStorageBytes	Number of noncurrent version bytes	Free	Cost optimization	N	-	Y

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% noncurrent version bytes	-	Percentage of bytes in scope that are noncurrent versions	Free	Cost opti ion	Y	sum rren ionS eByt s um(S eByt	Y
Noncurrent version object count	NonCurren tVersionO bjectCount	Count of the noncurrent object versions, inclusive of noncurrent delete markers	Free	Cost opti ion	N	-	Y
% noncurren t version objects	-	Percentage of objects in scope that are a noncurrent version	Free	Cost opti ion	Y	sum rren ionC Cou su m(O ount	Y
Delete marker bytes	DeleteMar kerStorag eBytes	Number of bytes in scope that are delete markers	Free	Cost opti ion	N	-	Y

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% delete marker bytes	-	Percentage of bytes in scope that are delete markers	Free	Cost opti ion	Y	sum eMa orag s)/ sum orag s)	Y
Delete marker object count	DeleteMar kerObject Count	Count of objects with a delete marker, both current and noncurrent	Free	Cost opti ion	N	-	Y
% delete marker objects	-	Percentage of objects in scope with a delete marker	Free	Cost opti ion	Y	sum eMa ject( )/ sum ectC	Y
Incomplet e multipart upload bytes	Incomplet eMultipar tUploadSt orageBytes	Total bytes in scope for incomplet e multipart uploads	Free	Cost opti ion	N	-	Y

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% incomplete multipart upload bytes	-	Percentage of bytes in scope that are the result of incomplete multipart uploads	Free	Cost optimization	Y	sum pleteipartadStByte sum(StByte	Y
Incomplete multipart upload object count	IncompleteMultipartUploadObjectCount	Number of objects in scope that are incomplete multipart uploads	Free	Cost optimization	N	-	Y
% incomplete multipart upload objects	-	Percentage of objects in scope that are incomplete multipart uploads	Free	Cost optimization	Y	sum pleteipartadOountsum (Objunt)	Y
Incomplete multipart upload storage bytes greater than 7 days old	IncompleteMPUStorageBytesOlderThan7Days	Total bytes in scope for incomplete multipart uploads that are more than 7 days old	Free	Cost optimization	N	-	Y

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% incomplete multipart upload storage bytes greater than 7 days old	-	Percentage of bytes for incomplete multipart uploads that are more than 7 days old	Free	Cost optimization	Y	sumpletoraesOlan7Isumgeby	Y
Incomplete multipart upload object count greater than 7 days old	IncompleteMPUObjectCountOlderThan7Days	Number of objects that are incomplete multipart uploads more than 7 days old	Free	Cost optimization	N	-	Y
% incomplete multipart upload object count greater than 7 days old	-	Percentage of objects that are incomplete multipart uploads more than 7 days old	Free	Cost optimization	Y	sumpletobjectOldn7Dsum(Cou	Y
Transition lifecycle rule count	TransitionLifecycleRuleCount	Number of lifecycle rules to transition objects to another storage class	Adva	Cost optimization	N	-	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
Average transition lifecycle rules per bucket	-	Average number of lifecycle rules to transitio n objects to another storage class	Adva	Cost opti ion	Y	sum ition cycle Cou su m(D tNur Buck	N
Expiration lifecycle rule count	Expiratio nLifecycl eRuleCount	Number of lifecycle rules to expire objects	Adva	Cost opti ion	N	-	N
Average expiration lifecycle rules per bucket	-	Average number of lifecycle rules to expire objects	Adva	Cost opti ion	Y	sum atio cycle Cou su m(D tNur Buck	N
Noncurren t version transition lifecycle rule count	Noncurren tVersionT ransition Lifecycle RuleCount	Number of lifecycle rules to transitio n noncurrent object versions to another storage class	Adva	Cost opti ion	N	-	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
Average noncurrent version transition lifecycle rules per bucket	-	Average number of lifecycle rules to transition noncurrent object versions to another storage class	Adva	Cost optimization	Y	sum rrentionTtionlycleountsum (Distnumucke	N
Noncurrent version expiration lifecycle rule count	NoncurrentVersionExpirationLifecycleRuleCount	Number of lifecycle rules to expire noncurrent object versions	Adva	Cost optimization	N	-	N
Average noncurrent version expiration lifecycle rules per bucket	-	Average number of lifecycle rules to expire noncurrent object versions	Adva	Cost optimization	Y	sum rrentionEtionlycleountsum (Distnumucke	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
Abort incomplete multipart upload lifecycle rule count	AbortIncompleteMPULifecycleRuleCount	Number of lifecycle rules to delete incomplete multipart uploads	Adva	Cost optimization	N	-	N
Average abort incomplete multipart upload lifecycle rules per bucket	-	Average number of lifecycle rules to delete incomplete multipart uploads	Adva	Cost optimization	Y	sum IncoeMPycleCountsum (DistNumucke	N
Expired object delete marker lifecycle rule count	ExpiredObjectDeleteMarkerLifecycleRuleCount	Number of lifecycle rules to remove expired object delete markers	Adva	Cost optimization	N	-	N
Average expired object delete marker lifecycle rules per bucket	-	Average number of lifecycle rules to remove expired object delete markers	Adva	Cost optimization	Y	sum edOelete erLifleRunt)/ sum istin mbe kets	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
Total lifecycle rule count	TotalLife cycleRule Count	Number of lifecycle rules	Adva	Cost opti ion	N	-	N
Average lifecycle rule count per bucket	-	Average number of lifecycle rules	Adva	Cost opti ion	Y	sum Lifec Rule)/ sum tinct erOf ts)	N
Encrypted bytes	Encrypted StorageBytes	Number of encrypted bytes	Free	Data prot n	N	-	Y
% encrypted bytes	-	Percentage of total bytes that are encrypted	Free	Data prot n	Y	sum ptec geBy (sum ageB - sum eMa orag s))	Y
Encrypted object count	Encrypted ObjectCount	Number of objects that are encrypted	Free	Data prot n	N	-	Y

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% encrypted objects	-	Percentage of objects that are encrypted	Free	Data prot n	Y	sum ptec tCou (sum tCou - sum eMa ject( ))	Y
Unencrypted bytes	-	Number of bytes that are unencrypted	Free	Data prot n	Y	sum geBy - sum eMa orag s) - sum ptec geBy	Y

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% unencrypted bytes	-	Percentage of bytes that are unencrypted	Free	Data prot n	Y	sum rypt rage)/ (sum orag s) - sum eMa orag s))	Y
Unencrypted object count	-	Number of objects that are unencrypted	Free	Data prot n	Y	sum tCou - sum eMa ject( ) - sum ptec tCou	Y

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% unencrypted objects	-	Percentage of unencrypted objects	Free	Data prot n	Y	sum rypt ectC / (sum ectC - sum eMa ject( ))	Y
Replicated storage bytes source	ReplicatedStorageBytesSource	Number of bytes that are replicated from the source bucket	Free	Data prot n	N	-	Y
% replicated bytes source	-	Percentage of total bytes that are replicated from the source bucket	Free	Data prot n	Y	sum cate ageB ourc su m(St Byte	Y
Replicated object count source	ReplicatedObjectCountSource	Number of replicated objects from the source bucket	Free	Data prot n	N	-	Y

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% replicated objects source	-	Percentage of total objects that are replicated from the source bucket	Free	Data prot n	Y	sum cate ctCo urce sum (Obj unt)	Y
Replicated storage bytes destination	ReplicatedStorageBytes	Number of bytes that are replicated to the destination bucket	Free	Data prot n	N	-	N
% replicated bytes destination	-	Percentage of total bytes that are replicated to the destination bucket	Free	Data prot n	Y	sum cate ageB / sum ageB	Y
Replicated object count destination	ReplicatedObjectCount	Number of objects that are replicated to the destination bucket	Free	Data prot n	N	-	Y
% replicated objects destination	-	Percentage of total objects that are replicated to the destination bucket	Free	Data prot n	Y	sum cate ctCo sum tCou	Y

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
Object Lock bytes	ObjectLockEnabledStorageBytes	Number of Object Lock enabled storage bytes	Free	Data prot n	N	-	Y
% Object Lock bytes	-	Percentage of Object Lock enabled storage bytes	Free	Data prot n	Y	sum tLocledSeByt s um(eByt	Y
Object Lock object count	ObjectLockEnabledObjectCount	Number of Object Lock objects	Free	Data prot n	N	-	Y
% Object Lock objects	-	Percentage of total objects that have Object Lock enabled	Free	Data prot n	Y	sum tLocledCCou su m(Ount	Y
Versioning-enabled bucket count	VersioningEnabledBucketCount	Number of buckets that have S3 Versioning enabled	Free	Data prot n	N	-	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% versionin g-enabled buckets	-	Percentage of buckets that have S3 Versioning enabled	Free	Data prot n	Y	sum onin ledB Cou su m(D tNur Buck	N
MFA delete-en abled bucket count	MFADelete EnabledBu cketCount	Number of buckets that have MFA (multi-factor authentic ation) delete enabled	Free	Data prot n	N	-	N
% MFA delete-en abled buckets	-	Percentage of buckets that have MFA (multi-factor authentic ation) delete enabled	Free	Data prot n	Y	sum letel edBu ount sum (Dist Num ucke	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
SSE-KMS enabled bucket count	SSEKMSEnabledBucketCount	Number of buckets that use server-side encryption with AWS Key Management Service keys (SSE-KMS) for default bucket encryption	Free	Data prot n	N	-	N
% SSE-KMS enabled buckets	-	Percentage of buckets that SSE-KMS for default bucket encryption	Free	Data prot n	Y	sum SEna ucke t)/ sum stinc berC ets)	N
All unsupported signature requests	AllUnsupportedSignatureRequests	Total number of requests that use unsupported AWS signature versions	Adva	Data prot n	N	-	N
% all unsupported signature requests	-	Percentage of requests that use unsupported AWS signature versions	Adva	Data prot n	Y	sum supp Sign Requ / sum eque	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
All unsupported TLS requests	AllUnsupportedTLSRequests	Total number of requests that use unsupported Transport Layer Security (TLS) versions	Adva	Data prot n	N	-	N
% all unsupported TLS requests	-	Percentage of requests that use unsupported TLS versions	Adva	Data prot n	Y	sum supp TLSR ts)/ sum llRe s)	N
All SSE-KMS requests	AllSSEKMSRequests	Total number of requests that specify SSE-KMS	Adva	Data prot n	N	-	N
% all SSE-KMS requests	-	Percentage of requests that specify SSE-KMS	Adva	Data prot n	Y	sum EKM sts)/ sum AllR ts)	N
Same-Region Replication rule count	SameRegionReplicationRuleCount	Number of replication rules for Same-Region Replication (SRR)	Adva	Data prot n	N	-	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
Average Same-Region Replication rules per bucket	-	Average number of replication rules for SRR	Adva	Data prot n	Y	sum egio icati leCo sum nctN OfBu)	N
Cross-Region Replication rule count	CrossRegionReplicationRuleCount	Number of replication rules for Cross-Region Replication (CRR)	Adva	Data prot n	N	-	N
Average Cross-Region Replication rules per bucket	-	Average number of replication rules for CRR	Adva	Data prot n	Y	sum Regi licat uleC / sum inctI rOfE s)	N
Same-account replication rule count	SameAccountReplicationRuleCount	Number of replication rules for replication within the same account	Adva	Data prot n	N	-	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
Average same-account replication rules per bucket	-	Average number of replication rules for replication within the same account	Adva	Data prot n	Y	sum ccou licat uleC / sum inctI rOfE s)	N
Cross-account replication rule count	CrossAccountReplicationRuleCount	Number of replication rules for cross-account replication	Adva	Data prot n	N	-	N
Average cross-account replication rules per bucket	-	Average number of replication rules for cross-account replication	Adva	Data prot n	Y	sum Accc plica Rule)/ sum tinct erOf ts)	N
Invalid destination replication rule count	InvalidDestinationReplicationRuleCount	Number of replication rules with a replication destination that's not valid	Adva	Data prot n	N	-	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
Average invalid destinati on replicati on rules per bucket	-	Average number of replicati on rules with a replication destination that's not valid	Adva	Data prot n	Y	sum idDe tionI catic eCou s um(I ctNu fBuc	N
Total replicati on rule count	-	Total replication rule count	Adva	Data prot n	Y	sum idDe tionI catic eCou + sum Accc plica Rule) + sum ccou licat uleC	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
Average replication rule count per bucket	-	Average total replication rule count	Adva	Data prot n	Y	(sum lidD atio icati leCo + sum Accc plica Rule) + sum ccou licat uleC)/ sum tinct erOf ts)	N
Object Ownership bucket owner enforced bucket count	ObjectOwnershipBucketOwnerEnforcedBucketCount	Number of buckets that have access control lists (ACLs) disabled by using the bucket owner enforced setting for Object Ownership	Free	Acce mar t	N	-	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% Object Ownership bucket owner enforced buckets	-	Percentage of buckets that have ACLs disabled by using the bucket owner enforced setting for Object Ownership	Free	Acce mar t	Y	sum tOw pBu nerE edBu ount sum (Dist Num ucke	N
Object Ownership bucket owner preferred bucket count	ObjectOwn ershipBuc ketOwnerP referredB ucketCount	Number of buckets that use the bucket owner preferred setting for Object Ownership	Free	Acce mar t	N	-	N
% Object Ownership bucket owner preferred buckets	-	Percentage of buckets that use the bucket owner preferred setting for Object Ownership	Free	Acce mar t	Y	sum tOw pBu nerF redB Cou su m(D tNu Buck	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
Object Ownership object writer bucket count	ObjectOwnershipObjectWriterBucketCount	Number of buckets that use the object writer setting for Object Ownership	Free	Acce mar t	N	-	N
% Object Ownership object writer buckets	-	Percentage of buckets that use the object writer setting for Object Ownership	Free	Acce mar t	Y	sum tOw pOb iterE tCou s um(ctNu fBuc	N
Transfer Accelerat ion enabled bucket count	TransferA ccelerati onEnabled BucketCount	Number of buckets that have Transfer Acceleration enabled	Free	Perf ce	N	-	N
% Transfer Accelerat ion enabled buckets	-	Percentage of buckets that have Transfer Acceleration enabled	Free	Perf ce	Y	sum ferA ratic bled tCou s um(ctNu fBuc	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
Event Notification enabled bucket count	EventNotificationEnabledBucketCount	Number of buckets that have Event Notifications enabled	Free	Ever	N	-	N
% Event Notification enabled buckets	-	Percentage of buckets that have Event Notifications enabled	Free	Ever	Y	sum NotionEdBucunt) sum Distiumbcket	N
All requests	AllRequests	Total number of requests made	Adva	Acti	N	-	N
Get requests	GetRequests	Total number of GET requests made	Adva	Acti	N	-	N
Put requests	PutRequests	Total number of PUT requests made	Adva	Acti	N	-	N
Head requests	HeadRequests	Number of HEAD requests made	Adva	Acti	N	-	N
Delete requests	DeleteRequests	Number of DELETE requests made	Adva	Acti	N	-	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
List requests	ListRequests	Number of LIST requests made	Adva	Acti	N	-	N
Post requests	PostRequests	Number of POST requests made	Adva	Acti	N	-	N
Select requests	SelectRequests	Number of S3 Select requests	Adva	Acti	N	-	N
Select scanned bytes	SelectScannedBytes	Number of S3 Select bytes scanned	Adva	Acti	N	-	N
Select returned bytes	SelectReturnedBytes	Number of S3 Select bytes returned	Adva	Acti	N	-	N
Bytes downloaded	BytesDownloaded	Number of bytes downloaded	Adva	Acti	N	-	N
% retrieval rate	-	Percentage of bytes downloaded	Adva	Acti	Y	sum Dow d)/ sum orag s)	N
Bytes uploaded	BytesUploaded	Number of bytes uploaded	Adva	Acti	N	-	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% ingest ratio	-	Percentage of bytes uploaded	Adva	Acti	Y	sum Uplc / sum agel	N
4xx errors	4xxErrors	Number of HTTP 4xx status codes	Adva	Acti	N	-	N
5xx errors	5xxErrors	Number of HTTP 5xx status codes	Adva	Acti	N	-	N
Total errors	-	The sum of all 4xx and 5xx errors	Adva	Acti	Y	sum rors) + sum rors)	N
% error rate	-	Total number of 4xx and 5xx errors as a percentage of total requests	Adva	Acti	Y	sum Erro s um(/ uest	N
200 OK status count	200OKStat usCount	Number of 200 OK status codes	Adva	Deta stat code	N	-	N
% 200 OK status	-	Total number of 200 OK status codes as a percentage of total requests	Adva	Deta stat code	Y	sum Stat nt)/ sum llRe s)	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met forn	Storage Lens groups
206 Partial Content status count	206PartialContentStatusCount	Number of 206 Partial Content status codes	Adva	Deta stat code	N	-	N
% 206 Partial Content status	-	Number of 206 Partial Content status codes as a percentage of total requests	Adva	Deta stat code	Y	sum rtial entS Cou su m(A ests)	N
400 Bad Request error count	400BadRequestError Count	Number of 400 Bad Request status codes	Adva	Deta stat code	N	-	N
% 400 Bad Request errors	-	Number of 400 Bad Request status codes as a percentage of total requests	Adva	Deta stat code	Y	sum dRe rror( )/ sum Requ	N
403 Forbidden error count	403ForbiddenErrorCount	Number of 403 Forbidden status codes	Adva	Deta stat code	N	-	N
% 403 Forbidden errors	-	Number of 403 Forbidden status codes as a percentage of total requests	Adva	Deta stat code	Y	sum rbid rorC / sum eque	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
404 Not Found error count	404NotFoundErrorCount	Number of 404 Not Found status codes	Adva	Deta stat code	N	-	N
% 404 Not Found errors	-	Number of 404 Not Found status codes as a percentage of total requests	Adva	Deta stat code	Y	sum tFou orCo sum ques	N
500 Internal Server Error count	500InternalServerErrorCount	Number of 500 Internal Server Error status codes	Adva	Deta stat code	N	-	N
% 500 Internal Server Errors	-	Number of 500 Internal Server Error status codes as a percentage of total requests	Adva	Deta stat code	Y	sum tern verE ount sum (AllR sts)	N
503 Service Unavailable error count	503ServiceUnavailableErrorCount	Number of 503 Service Unavailable status codes	Adva	Deta stat code	N	-	N

Metric name	CloudWatch and export	Description	Tier	Cate	D	Deri met form	Storage Lens groups
% 503 Service Unavailable errors	-	Number of 503 Service Unavailable status codes as a percentage of total requests	Adva	Deta stat code	Y	sum rvice ailab rorC / sum eque	N

[1] All free tier storage metrics are available at the Storage Lens group level. Advanced tier metrics are not available at the Storage Lens group level.

[2] Rule count metrics and bucket settings metrics aren't available at the prefix level.

The following table shows the performance metrics available in S3 Storage Lens and their availability in CloudWatch:

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Ave Firs Byt Lat	AverageFirstByteLatency	Average per-request time between when an Amazon S3 bucket receives a complete request and when it starts returning the response, measured over	Advanced	Performance	N	-	N

Me nar	CloudWatch and export	Description	Tier	Categor	Derive	Derived metric formula	Storage Lens groups
		the past 24 hours					
Ave Tot Req Lat	AverageTo talRequestLatency	Average elapsed per-request time between the first byte received and the last byte sent to an Amazon S3 bucket, measured over the past 24 hours	Advanced	Performar ce	N	-	N
Rea 0KE req cou	Read0KBRe questCount*	Number of GetObject requests with data sizes of 0KB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Categor	Derive	Derived metric formula	Storage Lens groups
Rea 0KE to 128 req cou	Read0KBTo 128KBRequ estCount*	Number of GetObject requests with data sizes greater than 0KB and up to 128KB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N
Rea 128 to 256 req cou	Read128KB To256KBRe questCount*	Number of GetObject requests with data sizes greater than 128KB and up to 256KB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Categor	Derive	Derived metric formula	Storage Lens groups
Rea 256 to 512 req cou	Read256KB To512KBRe questCount*	Number of GetObject requests with data sizes greater than 256KB and up to 512KB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N
Rea 512 to 1M req cou	Read512KB To1MBRequ estCount*	Number of GetObject requests with data sizes greater than 512KB and up to 1MB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Categor	Derive	Derived metric formula	Storage Lens groups
Rea 1M to 2M req cou	Read1MBTo 2MBRequestCount*	Number of GetObject requests with data sizes greater than 1MB and up to 2MB, including both range-based requests and whole object requests	Advanced	Performan ce	N	-	N
Rea 2M to 4M req cou	Read2MBTo 4MBRequestCount*	Number of GetObject requests with data sizes greater than 2MB and up to 4MB, including both range-based requests and whole object requests	Advanced	Performan ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Rea 4M to 8M req cou	Read4MBTo 8MBRequestCount*	Number of GetObject requests with data sizes greater than 4MB and up to 8MB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N
Rea 8M to 16M req cou	Read8MBTo 16MBReque stCount*	Number of GetObject requests with data sizes greater than 8MB and up to 16MB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Categor	Derive	Derived metric formula	Storage Lens groups
Rea 16M to 32M req cou	Read16MBTo32MBRequestCount*	Number of GetObject requests with data sizes greater than 16MB and up to 32MB, including both range-based requests and whole object requests	Advanced	Performance	N	-	N
Rea 32M to 64M req cou	Read32MBTo64MBRequestCount*	Number of GetObject requests with data sizes greater than 32MB and up to 64MB, including both range-based requests and whole object requests	Advanced	Performance	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Rea 64M to 128 req cou	Read64MBTo128MBRequestCount*	Number of GetObject requests with data sizes greater than 64MB and up to 128MB, including both range-based requests and whole object requests	Advanced	Performance	N	-	N
Rea 128 to 256 req cou	Read128MBTo256MBRequestCount*	Number of GetObject requests with data sizes greater than 128MB and up to 256MB, including both range-based requests and whole object requests	Advanced	Performance	N	-	N

Me nar	CloudWatch and export	Description	Tier	Categor	Derive	Derived metric formula	Storage Lens groups
Rea 256 to 512 req cou	Read256MB To512MBRe questCount*	Number of GetObject requests with data sizes greater than 256MB and up to 512MB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N
Rea 512 to 1GI req cou	Read512MB To1GBRequ estCount*	Number of GetObject requests with data sizes greater than 512MB and up to 1GB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Categor	Derive	Derived metric formula	Storage Lens groups
Rea 1GI to 2GI req cou	Read1GBTo 2GBRequestCount*	Number of GetObject requests with data sizes greater than 1GB and up to 2GB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N
Rea 2GI to 4GI req cou	Read2GBTo 4GBRequestCount*	Number of GetObject requests with data sizes greater than 2GB and up to 4GB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Rea 4GI + req cou	ReadLarge rThan4GBR equestCount*	Number of GetObject requests with data sizes greater than 4GB, including both range-based requests and whole object requests	Advanced	Performar ce	N	-	N
Wri 0KI req cou	Write0KBR equestCount*	Number of PutObject, UploadPart, and CreateMul tipartUpload requests with data sizes of 0KB	Advanced	Performar ce	N	-	N
Wri 0KI to 128 req cou	Write0KBT o128KBReq uestCount*	Number of PutObject, UploadPart, and CreateMul tipartUpload requests with data sizes greater than 0KB and up to 128KB	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Wri 128 to 256 req cou	Write128KBTo256KBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 128KB and up to 256KB	Advanced	Performance	N	-	N
Wri 256 to 512 req cou	Write256KBTo512KBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 256KB and up to 512KB	Advanced	Performance	N	-	N
Wri 512 to 1M req cou	Write512KBTo1MBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 512KB and up to 1MB	Advanced	Performance	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Wri 1M to 2M req cou	Write1MBTo2MBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 1MB and up to 2MB	Advanced	Performance	N	-	N
Wri 2M to 4M req cou	Write2MBTo4MBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 2MB and up to 4MB	Advanced	Performance	N	-	N
Wri 4M to 8M req cou	Write4MBTo8MBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 4MB and up to 8MB	Advanced	Performance	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Wri 8M to 16M req cou	Write8MBTo16MBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 8MB and up to 16MB	Advanced	Performance	N	-	N
Wri 16M to 32M req cou	Write16MBTo32MBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 16MB and up to 32MB	Advanced	Performance	N	-	N
Wri 32M to 64M req cou	Write32MBTo64MBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 32MB and up to 64MB	Advanced	Performance	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Wri 64M to 128 req cou	Write64MB To128MBRe questCount*	Number of PutObject, UploadPart, and CreateMul tipartUpload requests with data sizes greater than 64MB and up to 128MB	Advanced	Performar ce	N	-	N
Wri 128 to 256 req cou	Write128M BTo256MBR equestCount*	Number of PutObject, UploadPart, and CreateMul tipartUpload requests with data sizes greater than 128MB and up to 256MB	Advanced	Performar ce	N	-	N
Wri 256 to 512 req cou	Write256M BTo512MBR equestCount*	Number of PutObject, UploadPart, and CreateMul tipartUpload requests with data sizes greater than 256MB and up to 512MB	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Wri 512 to 1GI req cou	Write512MBTo1GBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 512MB and up to 1GB	Advanced	Performance	N	-	N
Wri 1GI to 2GI req cou	Write1GBTo2GBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 1GB and up to 2GB	Advanced	Performance	N	-	N
Wri 2GI to 4GI req cou	Write2GBTo4GBRequestCount*	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 2GB and up to 4GB	Advanced	Performance	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Wri 4GI + req cou	WriteLarg erThan4GB RequestCount*	Number of PutObject, UploadPart, and CreateMul tipartUpload requests with data sizes greater than 4GB	Advanced	Performar ce	N	-	N
Obj 0KI cou	Object0KBCount	Number of objects with size equal to 0KB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Obj 0KE to 128 cou	Object0KB To128KBCount	Number of objects with size greater than 0KB and less than or equal to 128KB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performan ce	N	-	N
Obj 128 to 256 cou	Object128 KBTo256KBCount	Number of objects with size greater than 128KB and less than or equal to 256KB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performan ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Obj 256 to 512 cou	Object256 KBTo512KBCount	Number of objects with size greater than 256KB and less than or equal to 512KB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performan ce	N	-	N
Obj 512 to 1M cou	Object512 KBTo1MBCount	Number of objects with size greater than 512KB and less than or equal to 1MB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performan ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Ob 1M to 2M cou	Object1MB To2MBCount	Number of objects with size greater than 1MB and less than or equal to 2MB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performan ce	N	-	N
Ob 2M to 4M cou	Object2MB To4MBCount	Number of objects with size greater than 2MB and less than or equal to 4MB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performan ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Ob 4M to 8M cou	Object4MB To8MBCount	Number of objects with size greater than 4MB and less than or equal to 8MB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performar ce	N	-	N
Ob 8M to 16M cou	Object8MB To16MBCount	Number of objects with size greater than 8MB and less than or equal to 16MB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Ob 16M to 32M cou	Object16MBTo32MBCount	Number of objects with size greater than 16MB and less than or equal to 32MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers	Advanced	Performance	N	-	N
Ob 32M to 64M cou	Object32MBTo64MBCount	Number of objects with size greater than 32MB and less than or equal to 64MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers	Advanced	Performance	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Obj 64M to 128 cou	Object64MBTo128MBCount	Number of objects with size greater than 64MB and less than or equal to 128MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers	Advanced	Performance	N	-	N
Obj 128 to 256 cou	Object128MBTo256MBCount	Number of objects with size greater than 128MB and less than or equal to 256MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers	Advanced	Performance	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Obj 256 to 512 cou	Object256 MBTo512MBCount	Number of objects with size greater than 256MB and less than or equal to 512MB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performan ce	N	-	N
Obj 512 to 1GI cou	Object512 MBTo1GBCount	Number of objects with size greater than 512MB and less than or equal to 1GB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performan ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Ob 1GI to 2GI cou	Object1GB To2GBCount	Number of objects with size greater than 1GB and less than or equal to 2GB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performan ce	N	-	N
Ob 2GI to 4GI cou	Object2GB To4GBCount	Number of objects with size greater than 2GB and less than or equal to 4GB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performan ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Ob 4GI + cou	ObjectLar gerThan4GBCount	Number of objects with size greater than 4GB, including current version, noncurren t versions, incomplet e multipart uploads, and delete markers	Advanced	Performar ce	N	-	N
Cor t Put 503 erro cou	Concurren tPut503ErrorCount	Number of 503 errors that are generated due to concurren t writes to the same object	Advanced	Performar ce	N	-	N
% Cor t Put 503 erro	-	Percentag e of 503 errors that are generated due to concurren t writes to the same object	Advanced	Performar ce	Y	100 * Concurren tPut503Er rorCount / AllRequests	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
Cro Reg ion req cou	CrossRegi onRequestCount	Number of requests that originate from a client in different Region than bucket's home Region	Advanced	Performar ce	N	-	N
% Cro Reg ion req	-	Percentage of requests that originate from a client in different Region than bucket's home Region	Advanced	Performar ce	Y	100 * CrossRegi onRequest Count / AllRequests	N
Cro Reg ion tra ed byt	CrossRegionTransfe rredBytes	Number of bytes that are transferr ed from calls in different Region than bucket's home Region	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Categor	Derive	Derived metric formula	Storage Lens groups
% Cro Reg ion tra ed byt	-	Percentag e of bytes transferred that originate from calls in different Region than bucket's home Region	Advanced	Performar ce	Y	100 * CrossRegi onTransfe rredBytes / (BytesDow nloaded + BytesUplo aded)	N
Cro Reg ion wit rep on req cou	CrossRegi onWithoutReplicati onRequestCount	Number of requests that originate from a client in different Region than bucket's home Region, excluding cross-region replication requests	Advanced	Performar ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Categor	Derive	Derived metric formula	Storage Lens groups
% Cro Reg ion wit rep on req	-	Percentage of requests that originate from a client in different Region than bucket's home Region, excluding cross-region replication requests	Advanced	Performan ce	Y	100 * CrossRegi onWithout Replicati onRequest Count / AllRequests	N
Cro Reg ion wit rep on tra ed byt	CrossRegi onWithoutReplicati onTransferredBytes	Number of bytes that are transferr ed from calls in different Region than bucket's home Region, excluding cross-region replication bytes	Advanced	Performan ce	N	-	N

Me nar	CloudWatch and export	Description	Tier	Categor	Derive	Derived metric formula	Storage Lens groups
% Cro Reg ion wit rep on tra ed byt	-	Percentag e of bytes transferred that originate from calls in different Region than bucket's home Region, excluding cross-region replication bytes	Advanced	Performar ce	Y	100 * CrossRegi onWithout Replicati onTransfe rredBytes / (BytesDow nloaded + BytesUplo aded)	N
In- Reg req cou	InRegionR equestCount	Number of requests that originate from a client in same Region as bucket's home Region	Advanced	Performar ce	N	-	N
% In- Reg req	-	Percentage of requests that originate from a client in same Region as bucket's home Region	Advanced	Performar ce	Y	100 * InRegionR equestCount / AllRequests	N

Me nar	CloudWatch and export	Description	Tier	Category	Derive	Derived metric formula	Storage Lens groups
In-Reg trai ed byt	InRegionTransferredBytes	Number of bytes that are transferred from calls from same Region as bucket's home Region	Advanced	Performance	N	-	N
% In-Reg trai ed byt	-	Percentage of bytes transferred that originate from calls from same Region as bucket's home Region	Advanced	Performance	Y	100 * InRegionTransferredBytes / (BytesDownloaded + BytesUploaded)	N
Uni obj acc cou dai	UniqueObjectsAccessedDailyCount	Number of objects that were accessed at least once in last 24 hrs	Advanced	Performance	N	-	N
% Uni obj acc cou dai	-	Percentage of objects that were accessed at least once in last 24 hrs	Advanced	Performance	Y	100 * UniqueObjectsAccessedDailyCount / ObjectCount	N

Setting Amazon S3 Storage Lens permissions

Amazon S3 Storage Lens requires new permissions in AWS Identity and Access Management (IAM) to authorize access to S3 Storage Lens actions. To grant these permissions, you can use an identity-based IAM policy. You can attach this policy to IAM users, groups, or roles to grant them permissions. Such permissions can include the ability to enable or disable S3 Storage Lens, or to access any S3 Storage Lens dashboard or configuration.

The IAM user or role must belong to the account that created or owns the dashboard or configuration, unless both of the following conditions are true:

- Your account is a member of AWS Organizations.
- You were given access to create organization-level dashboards by your management account as a delegated administrator.

Note

- You can't use your account's root user credentials to view Amazon S3 Storage Lens dashboards. To access S3 Storage Lens dashboards, you must grant the required IAM permissions to a new or existing IAM user. Then, sign in with those user credentials to access S3 Storage Lens dashboards. For more information, see Security best practices in IAM in the *IAM User Guide*.
- Using S3 Storage Lens on the Amazon S3 console can require multiple permissions. For example, to edit a dashboard on the console, you need the following permissions:
  - `s3:ListStorageLensConfigurations`
  - `s3:GetStorageLensConfiguration`
  - `s3:PutStorageLensConfiguration`

Topics

- Setting account permissions to use S3 Storage Lens
- Setting account permissions to use S3 Storage Lens groups
- Setting permissions to use S3 Storage Lens with AWS Organizations

Setting account permissions to use S3 Storage Lens

To create and manage S3 Storage Lens dashboards and Storage Lens dashboard configurations, you must have the following permissions, depending on which actions you want to perform:

The following table shows Amazon S3 Storage Lens related IAM permissions.

Action	IAM permissions

Create or update an S3 Storage Lens dashboard in the Amazon S3 console.	`s3:ListStorageLensConfigurations`
	`s3:GetStorageLensConfiguration`
	`s3:GetStorageLensConfigurationTagging`
	`s3:PutStorageLensConfiguration`
	`s3:PutStorageLensConfigurationTagging`

Get the tags of an S3 Storage Lens dashboard on the Amazon S3 console.	`s3:ListStorageLensConfigurations` `s3:GetStorageLensConfigurationTagging`

View an S3 Storage Lens dashboard on the Amazon S3 console.	`s3:ListStorageLensConfigurations`
	`s3:GetStorageLensConfiguration`
	`s3:GetStorageLensDashboard`

Delete an S3 Storage Lens dashboard on Amazon S3 console.	`s3:ListStorageLensConfigurations`
	`s3:GetStorageLensConfiguration`
	`s3:DeleteStorageLensConfiguration`

Create or update an S3 Storage Lens configuration by using the AWS CLI or an AWS SDK.	`s3:PutStorageLensConfiguration`

Action	IAM permissions
	`s3:PutStorageLensConfigurationTagging`
Get the tags of an S3 Storage Lens configuration by using the AWS CLI or an AWS SDK.	`s3:GetStorageLensConfigurationTagging`
View an S3 Storage Lens configuration by using the AWS CLI or an AWS SDK.	`s3:GetStorageLensConfiguration`
Delete an S3 Storage Lens configuration by using the AWS CLI or AWS SDK.	`s3:DeleteStorageLensConfiguration`

Note

- You can use resource tags in an IAM policy to manage permissions.
- An IAM user or role with these permissions can see metrics from buckets and prefixes that they might not have direct permission to read or list objects from.
- For S3 Storage Lens dashboards with prefix-level metrics enabled, if a selected prefix path matches with an object key, the dashboard might display the object key as another prefix.
- For metrics exports, which are stored in a bucket in your account, permissions are granted by using the existing `s3:GetObject` permission in the IAM policy. Similarly, for an AWS Organizations entity, the organization's management account or delegated administrator accounts can use IAM policies to manage access permissions for organization-level dashboard and configurations.

Setting account permissions to use S3 Storage Lens groups

You can use S3 Storage Lens groups to understand the distribution of your storage within buckets based on prefix, suffix, object tag, object size, or object age. You can attach Storage Lens groups to your dashboards to view their aggregated metrics.

To work with Storage Lens groups, you need certain permissions. For more information, see the section called “Storage Lens groups permissions”.

Setting permissions to use S3 Storage Lens with AWS Organizations

You can use Amazon S3 Storage Lens to collect storage metrics and usage data for all accounts that are part of your AWS Organizations hierarchy. The following table shows the actions and permissions related to using S3 Storage Lens with Organizations.

Action	IAM Permissions
Enable trusted access for S3 Storage Lens for your organization.	`organizations:EnableAWSServiceAccess`
Disable trusted access for S3 Storage Lens for your organization.	`organizations:DisableAWSServiceAccess`
Register a delegated administrator to create S3 Storage Lens dashboards or configurations for your organization.	`organizations:RegisterDelegatedAdministrator`
Deregister a delegated administrator so that they can no longer create S3 Storage Lens dashboards or configurations for your organization.	`organizations:DeregisterDelegatedAdministrator`

Additional permissions to create S3 Storage Lens organization-wide configurations.	`organizations:DescribeOrganization`
	`organizations:ListAccounts`
	`organizations:ListAWSServiceAccessForOrganization`
	`organizations:ListDelegatedAdministrators`
	`iam:CreateServiceLinkedRole`

Working with Amazon S3 Storage Lens by using the console and API

Amazon S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. You can use S3 Storage Lens metrics to generate summary insights, such as finding out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use S3 Storage Lens metrics to identify cost-optimization opportunities, implement data-protection and security best practices, and improve the performance of application workloads. For example, you can identify buckets that don't have S3 Lifecycle rules to expire incomplete multipart uploads that are more than 7 days old. You can also identify buckets that aren't following data-protection best practices, such as using S3 Replication or S3 Versioning. S3 Storage Lens also analyzes metrics to deliver contextual recommendations that you can use to optimize storage costs and apply best practices for protecting your data.

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics report in CSV or Parquet format to a general purpose S3 bucket or export the metrics directly to an AWS-managed S3 table bucket.

> **Note**
>
> Storage Lens only aggregates metrics for S3 general purpose buckets.

The following sections contain examples of creating, updating, and viewing S3 Storage Lens configurations and performing operations related to the feature. If you are using S3 Storage Lens with AWS Organizations, these examples also cover those use cases. In the examples, replace any placeholder values.

Topics

- Create an Amazon S3 Storage Lens dashboard
- Update an Amazon S3 Storage Lens dashboard
- Disable an Amazon S3 Storage Lens dashboard

- Delete an Amazon S3 Storage Lens dashboard
- List Amazon S3 Storage Lens dashboards
- View an Amazon S3 Storage Lens dashboard configuration details
- Managing AWS resource tags with S3 Storage Lens
- Helper files for using Amazon S3 Storage Lens

Create an Amazon S3 Storage Lens dashboard

You can create additional S3 Storage Lens custom dashboards that can be scoped to your organization in AWS Organizations or to specific AWS Regions or buckets within an account.

> **Note**
>
> Any updates to your dashboard configuration can take up to 48 hours to accurately display or visualize.

Using the S3 console

Use the following steps to create an Amazon S3 Storage Lens dashboard on the Amazon S3 console.

Step 1: Configure general settings

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to switch to.
3. In the left navigation pane, under **S3 Storage Lens**, choose **Dashboards**.
4. Choose **Create dashboard**.
5. On the **Dashboard** page, in the **General** section, do the following:
    a. View the **Home Region** for your dashboard. The home Region is the AWS Region where the configuration and metrics for this Storage Lens dashboard are stored.
    b. Enter a dashboard name.

Dashboard names must be fewer than 65 characters and must not contain special characters or spaces.

> **Note**
>
> You can't change this dashboard name after the dashboard is created.

c. Choose **Enabled** to display updated daily metrics in your dashboard.

d. (Optional) You can choose to add **Tags** to your dashboard. You can use tags to manage permissions for your dashboard and track costs for S3 Storage Lens. For more information, see Controlling access to AWS resources using tags in the *IAM User Guide* and Using AWS-generated tags in the *AWS Billing User Guide*.

> **Note**
>
> You can add up to 50 tags to your dashboard configuration.

6. Choose **Next** to save your changes and proceed.

Step 2: Define the dashboard scope

1. In the **Dashboard scope** section, choose the Regions and buckets that you want S3 Storage Lens to include or exclude in the dashboard.
2. Choose the buckets in your selected Regions that you want S3 Storage Lens to include or exclude. You can either include or exclude buckets, but not both. This option isn't available when you create organization-level dashboards.

> **Note**
>
> - You can either include or exclude Regions and buckets. This option is limited to Regions only when creating organization-level dashboards across member accounts in your organization.
> - You can choose up to 50 buckets to include or exclude.

3. Choose **Next** to save your changes and proceed.

Step 3: Choose your Storage Lens tier

1. In the **Storage Lens tier** section, choose the tier of features that you want to aggregate for this dashboard.

   a. To include free metrics aggregated at the bucket level and available for queries for 14 days, choose **Free tier**.

   b. To enable advanced metrics, choose **Advanced tier**. These options include prefix or Storage Lens groups aggregation, Amazon CloudWatch publishing, the expanded prefixes report, and contextual recommendations. Data is available for queries for 15 months. Advanced metrics and recommendations have an additional cost. For more information, see Amazon S3 pricing.

   For more information about advanced metrics and free metrics, see Metrics selection.

2. Under **Advanced metric categories**, select the category of metrics that you want to enable:

   - **Activity metrics**
   - **Detailed status code metrics**
   - **Cost optimization metrics**
   - **Data protection metrics**
   - **Performance metrics**

   To preview which metrics are included in each category, use the drop-down arrow button below the metrics category checkbox list. For more information about metrics categories, see Metrics categories. For a complete list of metrics, see Amazon S3 Storage Lens metrics glossary.

3. Choose or specify a **Prefix delimiter** to distinguish levels within each prefix. This value is used to identify each prefix level. The default value in Amazon S3 is the "/" character, but your storage structure might use other delimiter characters.

4. Choose **Next** to save your changes and proceed.

Step 4: (Optional) Choose your metrics aggregation

1. Under **Additional metrics aggregation**, choose which metrics you want to aggregate:

   - Prefix aggregation

- Storage Lens group aggregation

2. If you've enabled **Prefix aggregation**, specify the minimum **Prefix threshold** for your dashboard and **Prefix depth**. Then, choose **Next** to save and proceed.

> **Note**
>
> The **Prefix depth** setting determines how many hierarchical levels deep S3 Storage Lens will analyze your object prefixes, with a maximum limit of 10 levels. The **Prefix threshold** specifies the minimum percentage of total storage that a prefix must represent before it's included in Storage Lens metrics.

3. If you've enabled **Storage Lens group aggregation**, choose one of the following:

- **Include Storage Lens groups**
- **Exclude Storage Lens groups**

4. When you include Storage Lens groups in your aggregation, you can either **Include all Storage Lens groups in your home Region** or specify Storage Lens groups to include.
5. Choose **Next** to save your changes and proceed.

Step 5: (Optional) Choose your metrics export and publishing settings

1. Under **Metrics publishing**, choose **CloudWatch publishing** if you want to access your Storage Lens metrics in your CloudWatch dashboard.

> **Note**
>
> Prefix-level metrics aren't available in CloudWatch.

2. Under **Metrics export**, choose which Storage Lens dashboard data you want exported daily:

- **Default metrics report**
- **Expanded prefixes metrics report**

3. (Optional) If you chose **Default metrics report**, in the **Default metrics report** settings, choose the bucket type. You can export the report to either a general purpose Amazon S3 bucket or AWS-managed S3 table bucket. Based on the selected bucket type, update the **General purpose bucket destination settings** or **Table bucket destination settings** options.

> **Note**
>
> The **default metrics report** only includes prefixes within the set threshold and depth set in prefix aggregation settings.
> If you choose to specify an encryption key, you must choose an AWS KMS key (SSE-KMS) or Amazon S3 managed key (SSE-S3). If your destination bucket policy requires encryption, you must provide an encryption key for your metrics export. Without the encryption key, the export to S3 fails. For more information, see Using an AWS KMS key to encrypt your metrics exports.

4. (Optional) If you chose **Expanded prefixes metrics report**, in the **Expanded prefixes metrics report** settings, choose the bucket type. You can export the report to either a general purpose Amazon S3 bucket or a read-only S3 table bucket. Based on the selected bucket type, update the **General purpose bucket destination settings** or **Table bucket destination settings**.

> **Note**
>
> The **Expanded prefixes metrics report** includes all prefixes up to prefix depth 50 in all selected buckets that are specified in your dashboard scope.
> If you choose to specify an encryption key, you must choose an AWS KMS key (SSE-KMS) or Amazon S3 managed key (SSE-S3). If your destination bucket policy requires encryption, you must provide an encryption key for your metrics export. Without the encryption key, the export to S3 fails. For more information, see Using an AWS KMS key to encrypt your metrics exports.

5. Choose **Next** to save your changes and proceed.
6. Review everything on the **Review and Create** page. If there are no additional changes, choose **Next** to save your changes and to create your dashboard.

Step 6: Review your dashboard configuration and create your dashboard

1. In the **General** section, review your settings. Choose **Edit** to make any changes.
2. In the **Dashboard scope** section, review your settings. Choose **Edit** to make any changes.
3. In the **Storage Lens tier** section, review your settings. Choose **Edit** to make any changes.
4. In the **Metrics aggregation** section, review your settings. Choose **Edit** to make any changes.

5. In the **Metrics export** section, review your settings. Choose **Edit** to make any changes.
6. After reviewing and confirming all your dashboard configuration settings, choose **Submit** to create your dashboard.

After you've successfully created your new Storage Lens dashboard, you can view your new dashboard listed under your Storage Lens **Dashboard** page.

Using the AWS CLI

Example

The following example command creates a Amazon S3 Storage Lens configuration with tags. To use these examples, replace the *user input placeholders* with your own information.

```
aws s3control put-storage-lens-configuration --account-id=111122223333 --
config-id=example-dashboard-configuration-id --region=us-east-1 --storage-lens-
configuration=file://./config.json --tags=file://./tags.json
```

Example

The following example command creates a Amazon S3 Storage Lens configuration without tags. To use these examples, replace the *user input placeholders* with your own information.

```
aws s3control put-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json
```

Using the AWS SDK for Java

Example– Create and update an Amazon S3 Storage Lens configuration

The following example creates and updates an Amazon S3 Storage Lens configuration in SDK for Java:

```
package aws.example.s3control;

import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.core.exception.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
```

```
import software.amazon.awssdk.services.s3control.model.AccountLevel;
import software.amazon.awssdk.services.s3control.model.ActivityMetrics;
import software.amazon.awssdk.services.s3control.model.AdvancedCostOptimizationMetrics;
import software.amazon.awssdk.services.s3control.model.AdvancedDataProtectionMetrics;
import software.amazon.awssdk.services.s3control.model.AdvancedPerformanceMetrics;
import software.amazon.awssdk.services.s3control.model.BucketLevel;
import software.amazon.awssdk.services.s3control.model.CloudWatchMetrics;
import software.amazon.awssdk.services.s3control.model.DetailedStatusCodesMetrics;
import software.amazon.awssdk.services.s3control.model.Format;
import software.amazon.awssdk.services.s3control.model.Include;
import software.amazon.awssdk.services.s3control.model.OutputSchemaVersion;
import software.amazon.awssdk.services.s3control.model.PrefixLevel;
import software.amazon.awssdk.services.s3control.model.PrefixLevelStorageMetrics;
import
 software.amazon.awssdk.services.s3control.model.PutStorageLensConfigurationRequest;
import software.amazon.awssdk.services.s3control.model.S3BucketDestination;
import software.amazon.awssdk.services.s3control.model.SSES3;
import software.amazon.awssdk.services.s3control.model.SelectionCriteria;
import software.amazon.awssdk.services.s3control.model.StorageLensAwsOrg;
import software.amazon.awssdk.services.s3control.model.StorageLensConfiguration;
import software.amazon.awssdk.services.s3control.model.StorageLensDataExport;
import software.amazon.awssdk.services.s3control.model.StorageLensDataExportEncryption;
import
 software.amazon.awssdk.services.s3control.model.StorageLensExpandedPrefixesDataExport;
import software.amazon.awssdk.services.s3control.model.StorageLensTableDestination;
import software.amazon.awssdk.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::destBucketName"; // The destination
 bucket for your metrics export must be in the same Region as your S3 Storage Lens
 configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
```

```
            SelectionCriteria selectionCriteria = SelectionCriteria.builder()
                    .delimiter("/")
                    .maxDepth(5)
                    .minStorageBytesPercentage(10.0)
                    .build();

            PrefixLevelStorageMetrics prefixStorageMetrics =
PrefixLevelStorageMetrics.builder()
                    .isEnabled(true)
                    .selectionCriteria(selectionCriteria)
                    .build();

            BucketLevel bucketLevel = BucketLevel.builder()
                    .activityMetrics(ActivityMetrics.builder().isEnabled(true).build())

.advancedCostOptimizationMetrics(AdvancedCostOptimizationMetrics.builder().isEnabled(true).bui

.advancedDataProtectionMetrics(AdvancedDataProtectionMetrics.builder().isEnabled(true).build()

.advancedPerformanceMetrics(AdvancedPerformanceMetrics.builder().isEnabled(true).build())

.detailedStatusCodesMetrics(DetailedStatusCodesMetrics.builder().isEnabled(true).build())

.prefixLevel(PrefixLevel.builder().storageMetrics(prefixStorageMetrics).build())
                    .build();

            AccountLevel accountLevel = AccountLevel.builder()
                    .activityMetrics(ActivityMetrics.builder().isEnabled(true).build())

.advancedCostOptimizationMetrics(AdvancedCostOptimizationMetrics.builder().isEnabled(true).bui

.advancedPerformanceMetrics(AdvancedPerformanceMetrics.builder().isEnabled(true).build())

.advancedDataProtectionMetrics(AdvancedDataProtectionMetrics.builder().isEnabled(true).build()

.detailedStatusCodesMetrics(DetailedStatusCodesMetrics.builder().isEnabled(true).build())
                    .bucketLevel(bucketLevel)
                    .build();

            Include include = Include.builder()
                    .buckets(Arrays.asList("arn:aws:s3:::bucketName"))
                    .regions(Arrays.asList("us-west-2"))
                    .build();
```

```
            StorageLensDataExportEncryption exportEncryption =
StorageLensDataExportEncryption.builder()
                    .sses3(SSES3.builder().build())
                    .build();

            S3BucketDestination s3BucketDestination = S3BucketDestination.builder()
                    .accountId(exportAccountId)
                    .arn(exportBucketArn)
                    .encryption(exportEncryption)
                    .format(exportFormat)
                    .outputSchemaVersion(OutputSchemaVersion.V_1)
                    .prefix("Prefix")
                    .build();

            StorageLensTableDestination s3TablesDestination =
StorageLensTableDestination.builder()
                    .encryption(exportEncryption)
                    .isEnabled(true)
                    .build();

            CloudWatchMetrics cloudWatchMetrics = CloudWatchMetrics.builder()
                    .isEnabled(true)
                    .build();

            StorageLensDataExport dataExport = StorageLensDataExport.builder()
                    .cloudWatchMetrics(cloudWatchMetrics)
                    .s3BucketDestination(s3BucketDestination)
                    .storageLensTableDestination(s3TablesDestination)
                    .build();

            StorageLensAwsOrg awsOrg = StorageLensAwsOrg.builder()
                    .arn(awsOrgARN)
                    .build();

            StorageLensExpandedPrefixesDataExport expandedPrefixesDataExport =
StorageLensExpandedPrefixesDataExport.builder()
                    .s3BucketDestination(s3BucketDestination)
                    .storageLensTableDestination(s3TablesDestination)
                    .build();

            StorageLensConfiguration configuration = StorageLensConfiguration.builder()
                    .id(configurationId)
                    .accountLevel(accountLevel)
                    .include(include)
```

```
                    .dataExport(dataExport)
                    .awsOrg(awsOrg)
                    .expandedPrefixesDataExport(expandedPrefixesDataExport)
                    .prefixDelimiter("/")
                    .isEnabled(true)
                    .build();

            List<StorageLensTag> tags = Arrays.asList(
                    StorageLensTag.builder().key("key-1").value("value-1").build(),
                    StorageLensTag.builder().key("key-2").value("value-2").build()
            );

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();


 s3ControlClient.putStorageLensConfiguration(PutStorageLensConfigurationRequest.builder()
                    .accountId(sourceAccountId)
                    .configId(configurationId)
                    .storageLensConfiguration(configuration)
                    .tags(tags)
                    .build()
            );

        } catch (AwsServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

For access to S3 Storage Lens groups or expanded prefixes, you must upgrade your dashboard to use the advanced tier. Additional charges apply. For more information about the free and advanced tiers, see Metrics selection. For more information about S3 Storage Lens groups, see the section called “Working with Storage Lens groups”.

Update an Amazon S3 Storage Lens dashboard

The Amazon S3 Storage Lens default dashboard is `default-account-dashboard`. This dashboard is preconfigured by Amazon S3 to help you visualize summarized insights and trends for your entire account's aggregated free and advanced metrics on the console. You can't modify the default dashboard's configuration scope, but you can upgrade the metrics selection from the free metrics to the paid advanced metrics and recommendations, configure the optional metrics export, or even disable the default dashboard. The default dashboard can't be deleted, and can only be disabled. For more information, see Using the S3 console.

Using the S3 console

Use the following steps to update an Amazon S3 Storage Lens dashboard on the Amazon S3 console.

Step 1: Update your dashboard and configure your general settings

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens, Dashboards**.
3. Choose the dashboard that you want to edit.
4. Choose **View dashboard configuration**.
5. Choose **Edit**. You can now review the dashboard configuration, step by step. To make changes to any of the steps, you can choose the step directly using the left navigation. For instructions on how to update those steps,

   > **Note**
   >
   > You can't change the following:
   >
   > - The dashboard name
   > - The home Region

6. On the **Dashboard** page, in the **General** section, you can make changes to the following:

   - Choose **Enabled** or **Disabled** to update whether you're receiving daily metrics in your dashboard.

- (Optional) You can choose to add **Tags** to your dashboard. You can use tags to manage permissions for your dashboard and track costs for S3 Storage Lens. For more information, see Controlling access to AWS resources using tags in the *IAM User Guide* and Using AWS-generated tags in the *AWS Billing User Guide*.

> **Note**
>
> You can add up to 50 tags to your dashboard configuration.

7. Choose **Next** to save your changes and proceed.

Step 2: Update the dashboard scope

1. In the **Dashboard scope** section, update the Regions and buckets that you want S3 Storage Lens to include or exclude in the dashboard.

> **Note**
>
> - You can either include or exclude Regions and buckets. This option is limited to Regions only when creating organization-level dashboards across member accounts in your organization.
> - You can choose up to 50 buckets to include or exclude.

2. Choose the buckets in your selected Regions that you want S3 Storage Lens to include or exclude. You can either include or exclude buckets, but not both. This option isn't available when you create organization-level dashboards.

> **Note**
>
> - You can either include or exclude Regions and buckets. This option is limited to Regions only when creating organization-level dashboards across member accounts in your organization.
> - You can choose up to 50 buckets to include or exclude.

3. Choose **Next** to save your changes and proceed.

Step 3: Update your Storage Lens tier Configure the metrics selection

1. In the **Storage Lens tier Metrics selection** section, update the tier of metrics that you want to aggregate for this dashboard.

   > **Note**
   >
   > - If you're updating from the **Free tier** to the **Advanced tier**, you'll need to update your **Metrics aggregation** settings. To update your **Metrics aggregation settings**, see **Step 4: Update your metrics aggregation**.
   > - If you're updating your Storage Lens tier from the **Advanced tier** to the **Free tier**, you won't need to update any **Metrics aggregation** settings. The **Metrics aggregation** feature only applies to **Advanced tier** metric categories.

2. To include free metrics aggregated at the bucket level and available for queries for 14 days, choose **Free tier**.
3. To enable advanced metrics, choose **Advanced tier**. These options include prefix aggregation, Amazon CloudWatch publishing, and contextual recommendations. Data is available for queries for 15 months. Advanced metrics and recommendations have an additional cost. For more information, see Amazon S3 pricing.

   For more information about advanced metrics and free metrics, see Metrics selection.
4. Under **Advanced metric categories**, choose the category of metrics that you want to enable:

   - **Activity metrics**
   - **Detailed status code metrics**
   - **Cost optimization metrics**
   - **Data protection metrics**
   - **Performance metrics**

   To preview which metrics are included in each category, use the drop-down arrow button below the metrics category checkbox list. For more information about metrics categories, see Metrics categories. For a complete list of metrics, see Amazon S3 Storage Lens metrics glossary.

5. Choose or specify a **Prefix delimiter** to distinguish levels within each prefix. This value is used to identify each prefix level. The default value in Amazon S3 is the "/" character, but your storage structure might use other delimiter characters.
6. Choose **Next** to save your changes and proceed.

Step 4: (Optional) Update your metrics aggregation

1. Under **Additional metrics aggregation**, update which metrics you want to aggregate by choosing one of the following:
   - Prefix aggregation
   - Storage Lens group aggregation
2. If you've enabled **Prefix aggregation**, specify the minimum **Prefix threshold** for your dashboard and **Prefix depth**. Then, choose **Next** to save and proceed.
3. If you've enabled **Storage Lens group aggregation**, choose one of the following:
   - **Include Storage Lens groups**
   - **Exclude Storage Lens groups**
4. When you include Storage Lens groups in your aggregation, you can either **Include all Storage Lens groups in your home Region** or specify Storage Lens groups to include.
5. Choose **Next** to save your changes and proceed.

Step 5: (Optional) Update your metrics export and publishing settings

1. Under **Metrics publishing**, choose **CloudWatch publishing** if you want to access your Storage Lens metrics in your CloudWatch dashboard.

   > **Note**
   >
   > Prefix-level metrics aren't available in CloudWatch.

2. Under **Metrics export**, choose which Storage Lens dashboard data you want exported daily:
   - **Default metrics report**
   - **Expanded prefixes metrics report**

3. (Optional) If you chose **Default metrics report**, in the **Default metrics report** settings, choose the bucket type. You can export the report to either a general purpose S3 bucket or a read-only S3 table bucket. Based on the selected bucket type, update the **General purpose bucket destination settings** or **Table bucket destination settings** options.

   Note

   - The **default metrics report** only includes prefixes within the set threshold and depth set in prefix aggregation settings. If your prefix aggregation isn't already configured, the threshold includes up to the 100 largest prefixes by size.
   - If you choose to specify an encryption key, you must choose an AWS KMS key (SSE-KMS) or Amazon S3 managed key (SSE-S3). If your destination bucket policy requires encryption, you must provide an encryption key for your metrics export. Without the encryption key, the export to S3 fails. For more information, see Using an AWS KMS key to encrypt your metrics exports.

4. Choose **Next** to save your changes and proceed.

5. (Optional) If you chose **Expanded prefixes metrics report**, in the **Expanded prefixes metrics report** settings, choose the bucket type. You can export the report to either a general purpose S3 bucket or a read-only S3 table bucket. Based on the selected bucket type, update the **General purpose bucket destination settings** or **Table bucket destination settings**.

   Note

   - The **Expanded prefixes metrics report** includes prefixes in all buckets that are specified in your dashboard scope.
   - If you choose to specify an encryption key, you must choose an AWS KMS key (SSE-KMS) or Amazon S3 managed key (SSE-S3). If your destination bucket policy requires encryption, you must provide an encryption key for your metrics export. Without the encryption key, the export to S3 fails. For more information, see Using an AWS KMS key to encrypt your metrics exports.

6. Choose **Next** to save your changes and proceed.

Step 6: Review and update your dashboard configuration

1. In the **General** section, review your settings. Choose **Edit** to make any changes.
2. In the **Dashboard scope** section, review your settings. Choose **Edit** to make any changes.
3. In the **Storage Lens tier** section, review your settings. Choose **Edit** to make any changes.
4. In the **Metrics aggregation** section, review your settings. Choose **Edit** to make any changes.
5. In the **Metrics export** section, review your settings. Choose **Edit** to make any changes.
6. After reviewing and confirming all your dashboard configuration settings, choose **Submit** to update your dashboard.

After you've successfully updated your new Storage Lens dashboard, you can view your updated dashboard configuration listed under your Storage Lens **Dashboard** page.

Using the AWS CLI

Example

The following example command updates a Amazon S3 Storage Lens dashboard configuration. To use these examples, replace the *`user input placeholders`* with your own information.

```
aws s3control put-storage-lens-configuration --account-id=111122223333 --
config-id=example-dashboard-configuration-id --region=us-east-1 --storage-lens-
configuration=file://./config.json --tags=file://./tags.json
```

Using the AWS SDK for Java

Example– Update a Amazon S3 Storage Lens configuration with advanced metrics and recommendations

The following examples shows you how to update the default S3 Storage Lens configuration with advanced metrics and recommendations in SDK for Java:

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
```

```
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class UpdateDefaultConfigWithPaidFeatures {

    public static void main(String[] args) {
        String configurationId = "default-account-dashboard"; // This configuration ID
 cannot be modified.
        String sourceAccountId = "111122223333";

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
```

```
                    .withBucketLevel(bucketLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );

        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

For access to S3 Storage Lens groups or expanded prefixes, you must upgrade your dashboard to use the advanced tier. Additional charges apply. For more information about the free and advanced tiers, see Metrics selection. For more information about S3 Storage Lens groups, see the section called “Working with Storage Lens groups”.

Disable an Amazon S3 Storage Lens dashboard

You can disable an Amazon S3 Storage Lens dashboard from the Amazon S3 console. Disabling a dashboard prevents it from generating metrics in the future. A disabled dashboard still retains its configuration information, so that it can be easily resumed when re-enabled. A disabled dashboard retains its historical data until it's no longer available for queries.

Using the S3 console

Use the following steps to disable an Amazon S3 Storage Lens dashboard on the Amazon S3 console.

To disable an Amazon S3 Storage Lens dashboard

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to disable, and then choose **Disable**.
4. On the confirmation page, confirm that you want to disable the dashboard by entering the name of dashboard in the text field, and then choose **Confirm**.

Delete an Amazon S3 Storage Lens dashboard

You can't delete the default dashboard. However, you can disable it. Before deleting a dashboard that you've created, consider the following:

- As an alternative to deleting a dashboard, you can *disable* the dashboard so that it is available to be re-enabled in the future. For more information, see Using the S3 console.
- Deleting the dashboard deletes all the configuration settings that are associated with it.
- Deleting a dashboard makes all the historic metrics data unavailable. This historical data is still retained for 15 months. If you want to access this data again, create a dashboard with the same name in the same home Region as the one that was deleted.

Using the S3 console

You can delete an Amazon S3 Storage Lens dashboard from the Amazon S3 console. However, deleting a dashboard prevents it from generating metrics in the future.

Deleting an Amazon S3 Storage Lens dashboard

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.

3. In the **Dashboards** list, choose the dashboard that you want to delete, and then choose **Delete**.
4. On the **Delete dashboards** page, confirm that you want to delete the dashboard by entering the name of dashboard in the text field. Then choose **Confirm**.

Using the AWS CLI

Example

The following example deletes a S3 Storage Lens configuration. To use these examples, replace the *`user input placeholders`* with your own information.

```
aws s3control delete-storage-lens-configuration --account-id=222222222222 --region=us-east-1 --config-id=your-configuration-id
```

Using the AWS SDK for Java

Example– Delete an Amazon S3 Storage Lens dashboard configuration

The following example shows you how to delete an S3 Storage Lens configuration using SDK for Java:

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class DeleteDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
```

```
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.deleteStorageLensConfiguration(new
 DeleteStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

List Amazon S3 Storage Lens dashboards

Using the S3 console

To list S3 Storage Lens dashboards

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, navigate to **Storage Lens**.
3. Choose **Dashboards**. You can now view the dashboards in your AWS account.

Using the AWS CLI

Example

The following example command lists the S3 Storage Lens dashboards in your AWS account. To use these examples, replace the *`user input placeholders`* with your own information.

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-
east-1 --next-token=abcdefghij1234
```

Example

The following example lists S3 Storage Lens configurations without a next token. To use these examples, replace the *user input placeholders* with your own information.

```
aws s3control list-storage-lens-configurations --account-id=222222222222 --region=us-east-1
```

Using the AWS SDK for Java

Example– List S3 Storage Lens dashboard configurations

The following examples shows you how to list S3 Storage Lens configurations in SDK for Java. To use this example, replace the *user input placeholders* with your own information." to each example description.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.ListStorageLensConfigurationEntry;
import com.amazonaws.services.s3control.model.ListStorageLensConfigurationsRequest;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class ListDashboard {

    public static void main(String[] args) {
        String sourceAccountId = "111122223333";
        String nextToken = "nextToken";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();
```

```
            final List<ListStorageLensConfigurationEntry> configurations =
                    s3ControlClient.listStorageLensConfigurations(new
 ListStorageLensConfigurationsRequest()
                            .withAccountId(sourceAccountId)
                            .withNextToken(nextToken)
                    ).getStorageLensConfigurationList();

            System.out.println(configurations.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

View an Amazon S3 Storage Lens dashboard configuration details

You can view a Amazon S3 Storage Lens dashboard from the Amazon S3 console, AWS CLI, and SDK for Java.

Using the S3 console

To view S3 Storage Lens dashboard configuration details

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. On the left navigation pane, navigate to **Storage Lens**.
3. Choose **Dashboards**.
4. From the **Dashboards** list, choose the dashboard that you want to view. You can now view the details of your Storage Lens dashboard.

Using the AWS CLI

Example

The following example retrieves an S3 Storage Lens configuration so that you can view the configuration details. To use these examples, replace the *`user input placeholders`* with your own information.

```
aws s3control get-storage-lens-configuration --account-id=222222222222 --config-
id=your-configuration-id --region=us-east-1
```

Using the AWS SDK for Java

Example– Retrieve and view an S3 Storage Lens configuration

The following example shows you how to retrieve an S3 Storage Lens configuration in SDK for Java so that you can view the configuration details. To use this example, replace the *`user input placeholders`* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.GetStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.GetStorageLensConfigurationResult;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class GetDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();
```

```
            final StorageLensConfiguration configuration =
                    s3ControlClient.getStorageLensConfiguration(new
 GetStorageLensConfigurationRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getStorageLensConfiguration();

            System.out.println(configuration.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Managing AWS resource tags with S3 Storage Lens

Each Amazon S3 Storage Lens dashboard is counted as an AWS resource with its own Amazon Resource Name (ARN). Therefore, when you configure your Storage Lens dashboard, you can optionally add AWS resource tags to the dashboard. You can add up to 50 tags for each Storage Lens dashboard. To create a Storage Lens dashboard with tags, you must have the following S3 Storage Lens permissions:

- `s3:ListStorageLensConfigurations`
- `s3:GetStorageLensConfiguration`
- `s3:GetStorageLensConfigurationTagging`
- `s3:PutStorageLensConfiguration`
- `s3:PutStorageLensConfigurationTagging`

You can use AWS resource tags to categorize resources according to department, line of business, or project. This is useful when you have many resources of the same type. By applying tags, you can quickly identify a specific S3 Storage Lens dashboard based on the tags that you've assigned to it. You can also use tags to track and allocate costs.

In addition, when you add an AWS resource tag to your Storage Lens dashboard, you activate attribute-based access control (ABAC). ABAC is an authorization strategy that defines permissions based on attributes such as tags. You can also use conditions that specify resource tags in your IAM policies to control access to AWS resources.

You can edit tag keys and values, and you can remove tags from a resource at any time. Also, be aware of the following limitations:

- Tag keys and tag values are case sensitive.
- If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value.
- If you delete a resource, any tags for the resource are also deleted.
- Don't include private or sensitive data in your AWS resource tags.
- System tags (with tag keys that begin with `aws:`) aren't supported.
- The length of each tag key can't exceed 128 characters. The length of each tag value can't exceed 256 characters.

The following examples demonstrate how to use AWS resource tags with Storage Lens dashboard.

Topics

- Add AWS resource tags to a Storage Lens dashboard
- Retrieve AWS resource tags for a Storage Lens dashboard
- Updating Storage Lens dashboard tags
- Deleting AWS resource tags from a S3 Storage Lens dashboard

Add AWS resource tags to a Storage Lens dashboard

The following examples demonstrate how to add AWS resource tags to an S3 Storage Lens dashboard. You can add resource tags by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To add AWS resource tags to a Storage Lens dashboard

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, navigate to **Storage Lens** on the left navigation panel.
3. Choose **Dashboards**.
4. Choose the radio button for the Storage Lens dashboard that you want to update. Then, choose **Edit**.
5. Under **General**, choose **Add tag**.
6. On the **Add tag** page, add the new key-value pair.

> **Note**
>
> Adding a new tag with the same key as an existing tag overwrites the previous tag value.

7. (Optional) To add more than one new tag, choose **Add tag** again to continue adding new entries. You can add up to 50 AWS resource tags to your Storage Lens dashboard.
8. (Optional) If you want to remove a newly added entry, choose **Remove** next to the tag that you want to remove.
9. Choose **Save changes**.

Using the AWS CLI

Example

The following example command adds tags to a S3 Storage Lens dashboard configuration. To use these examples, replace the *`user input placeholders`* with your own information.

```
aws s3control put-storage-lens-configuration-tagging --account-id=222222222222 --
region=us-east-1 --config-id=your-configuration-id --tags=file://./tags.json
```

Using the AWS SDK for Java

The following example adds tags to an Amazon S3 Storage Lens configuration in SDK for Java. To use this example, replace the *`user input placeholders`* with your own information.

Example– Add tags to an S3 Storage Lens configuration

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
```

```
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import
 com.amazonaws.services.s3control.model.PutStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class PutDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfigurationTagging(new
 PutStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
```

```
        }
    }
}
```

Retrieve AWS resource tags for a Storage Lens dashboard

The following examples demonstrate how to retrieve AWS resource tags for a S3 Storage Lens dashboard. You can get resource tags by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To retrieve the AWS resource tags for a Storage Lens dashboard

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, navigate to **Storage Lens**.
3. Choose **Dashboards**.
4. Choose the radio button for the Storage Lens dashboard configuration that you want to view. Then, choose **View dashboard configuration**.
5. Under **Tags**, review the tags associated with the dashboard.
6. (Optional) If you want to add a new tag, choose **Edit**. Then, choose **Add tag**. On the **Add tag** page, add the new key-value pair.

   > **Note**
   >
   > Adding a new tag with the same key as an existing tag overwrites the previous tag value.

7. (Optional) If you want to remove a newly added entry, choose **Remove** next to the tag that you want to remove.
8. Choose **Save changes**.

Using the AWS CLI

Example

The following example command retrieves tags for a S3 Storage Lens dashboard configuration. To use these examples, replace the *user input placeholders* with your own information.

```
aws s3control get-storage-lens-configuration-tagging --account-id=222222222222 --
region=us-east-1 --config-id=your-configuration-id --tags=file://./tags.json
```

Using the AWS SDK for Java

Example– Get tags for an S3 Storage Lens dashboard configuration

The following example shows you how to retrieve tags for an S3 Storage Lens dashboard configuration in SDK for Java. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationRequest;
import
 com.amazonaws.services.s3control.model.GetStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;

import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class GetDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
```

```
                    .withRegion(US_WEST_2)
                    .build();

            final List<StorageLensTag> s3Tags = s3ControlClient
                    .getStorageLensConfigurationTagging(new
 GetStorageLensConfigurationTaggingRequest()
                            .withAccountId(sourceAccountId)
                            .withConfigId(configurationId)
                    ).getTags();

            System.out.println(s3Tags.toString());
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Updating Storage Lens dashboard tags

The following examples demonstrate how to update Storage Lens dashboard tags by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To update an AWS resource tag for a Storage Lens dashboard

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, navigate to **Storage Lens**.
3. Choose **Dashboards**.
4. Choose the radio button for the Storage Lens dashboard configuration that you want to view. Then, choose **View dashboard configuration**.
5. Under **Tags**, review the tags associated with the dashboard.
6. (Optional) If you want to add a new tag, choose **Edit**. Then, choose **Add tag**. On the **Add tag** page, add the new key-value pair.

> **Note**
>
> Adding a new tag with the same key as an existing tag overwrites the previous tag value.

7. (Optional) If you want to remove a newly added entry, choose **Remove** next to the tag that you want to remove.
8. Choose **Save changes**.

Using the AWS CLI

Example

The following example command adds or replaces tags on an existing Amazon S3 Storage Lens dashboard configuration. To use these examples, replace the *`user input placeholders`* with your own information.

```
aws s3control put-storage-lens-configuration-tagging --account-id=111122223333 --
config-id=example-dashboard-configuration-id --region=us-east-1 --config-id=your-
configuration-id
```

Using the AWS SDK for Java

The following AWS SDK for Java example updates the AWS resource tags on an existing Storage Lens dashboard. To use this example, replace the *`user input placeholders`* with your own information.

Example– Update tags on an existing Storage Lens dashboard configuration

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import
 com.amazonaws.services.s3control.model.PutStorageLensConfigurationTaggingRequest;
import com.amazonaws.services.s3control.model.StorageLensTag;
```

```
import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class PutDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfigurationTagging(new
 PutStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Deleting AWS resource tags from a S3 Storage Lens dashboard

The following examples demonstrate how to delete AWS resource tags from an existing Storage Lens dashboard. You can delete tags by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To delete AWS resource tags from an existing Storage Lens dashboard

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, navigate to **Storage Lens**.
3. Choose **Dashboards**.
4. Choose the radio button for the Storage Lens dashboard configuration that you want to view. Then, choose **View dashboard configuration**.
5. Under **Tags**, review the tags associated with the dashboard.
6. Choose **Remove** next to the tag that you want to remove.
7. Choose **Save changes**.

Using the AWS CLI

The following AWS CLI command deletes AWS resource tags from an existing Storage Lens dashboard. To use this example command, replace the *`user input placeholders`* with your own information.

Example

```
aws s3control delete-storage-lens-configuration-tagging --account-id=222222222222 --
config-id=your-configuration-id --region=us-east-1
```

Using the AWS SDK for Java

The following AWS SDK for Java example deletes an AWS resource tag from the Storage Lens dashboard using the Amazon Resource Name (ARN) that you specify in account *`111122223333`*. To use this example, replace the *`user input placeholders`* with your own information.

Example– Delete tags for an S3 Storage Lens dashboard configuration

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import
 com.amazonaws.services.s3control.model.DeleteStorageLensConfigurationTaggingRequest;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class DeleteDashboardTagging {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "111122223333";
        try {
            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.deleteStorageLensConfigurationTagging(new
 DeleteStorageLensConfigurationTaggingRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Helper files for using Amazon S3 Storage Lens

Use the following JSON files and its key inputs for your examples.

S3 Storage Lens example configuration in JSON

Example `config.json`

The `config.json` file contains the details of a S3 Storage Lens Organizations-level *advanced metrics and recommendations* configuration. To use the following example, replace the *`user input placeholders`* with your own information.

> **Note**
>
> Additional charges apply for advanced metrics and recommendations. For more information, see advanced metrics and recommendations.

```
{
  "Id": "SampleS3StorageLensConfiguration", //Use this property to identify your S3
 Storage Lens configuration.
  "AwsOrg": { //Use this property when enabling S3 Storage Lens for AWS Organizations.
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true
      },
      "AdvancedDataProtectionMetrics": {
```

```
      "IsEnabled":true
      },
      "AdvancedCostOptimizationMetrics": {
        "IsEnabled":true
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true
      },
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true,
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": { //Replace with "Include" if you prefer to include Regions.
    "Regions": [
      "eu-west-1"
    ],
    "Buckets": [ //This attribute is not supported for AWS Organizations-level
 configurations.
      "arn:aws:s3:::amzn-s3-demo-source-bucket"
    ]
  },
  "IsEnabled": true, //Whether the configuration is enabled
  "DataExport": { //Details about the metrics export
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV", //You can add "Parquet" if you prefer.
      "AccountId": "111122223333",
      "Arn": "arn:aws:s3:::
amzn-s3-demo-destination-bucket", // The destination bucket for your metrics export
 must be in the same Region as your S3 Storage Lens configuration.
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
```

```
      "IsEnabled": true
    }
  }
}
```

S3 Storage Lens example configuration with Storage Lens groups in JSON

Example `config.json`

The `config.json` file contains the details that you want to apply to your Storage Lens configuration when using Storage Lens groups. To use the example, replace the *`user input placeholders`* with your own information.

To attach all Storage Lens groups to your dashboard, update your Storage Lens configuration with the following syntax:

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
    "StorageLensGroupLevel": {},
  "IsEnabled": true
}
```

To include only two Storage Lens groups in your Storage Lens dashboard configuration (*`slg-1`* and *`slg-2`*), use the following syntax:

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
```

```
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
        "SelectionCriteria": {
            "Include": [
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
    },
  "IsEnabled": true
}
```

To exclude only certain Storage Lens groups from being attached to your dashboard configuration, use the following syntax:

```
{
  "Id": "ExampleS3StorageLensConfiguration",
  "AccountLevel": {
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
      "IsEnabled":true
      },
   "StorageLensGroupLevel": {
```

```
        "SelectionCriteria": {
            "Exclude": [
                "arn:aws:s3:us-east-1:111122223333:storage-lens-group/slg-1",
                "arn:aws:s3:us-east-1:444455556666:storage-lens-group/slg-2"
            ]
    },
  "IsEnabled": true
}
```

S3 Storage Lens example tags configuration in JSON

Example `tags.json`

The `tags.json` file contains the tags that you want to apply to your S3 Storage Lens configuration. To use this example, replace the *`user input placeholders`* with your own information.

```
[
    {
        "Key": "key1",
        "Value": "value1"
    },
    {
        "Key": "key2",
        "Value": "value2"
    }
]
```

S3 Storage Lens example configuration IAM permissions

Example `permissions.json` – Specific dashboard name

This example policy shows an S3 Storage Lens IAM `permissions.json` file with a specific dashboard name specified. Replace *`value1`*, *`us-east-1`*, *`your-dashboard-name`*, and *`example-account-id`* with your own values.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetStorageLensConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
            "Resource": "arn:aws:s3:us-east-1:111122223333:storage-lens/your-
dashboard-name"
        }
    ]
}
```

Example `permissions.json` – No specific dashboard name

This example policy shows an S3 Storage Lens IAM `permissions.json` file without a specific dashboard name specified. Replace *value1*, *us-east-1*, and *example-account-id* with your own values.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetStorageLensConfiguration",
                "s3:DeleteStorageLensConfiguration",
                "s3:PutStorageLensConfiguration"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/key1": "value1"
                }
            },
```

```
            "Resource": "arn:aws:s3:us-east-1:111122223333:storage-lens/*"
        }
    ]
}
```

Viewing metrics with Amazon S3 Storage Lens

S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics report in CSV or Parquet format to a general purpose S3 bucket or export the metrics directly to an AWS-managed S3 table bucket.

By default, all dashboards are configured with free metrics, which include metrics that you can use to understand usage and activity across your S3 storage, optimize your storage costs, and implement data-protection and access-management best practices. Free metrics are aggregated down to the bucket level. With free metrics, data is available for queries for up to 14 days.

Advanced metrics and recommendations include the following additional features that you can use to gain further insight into usage and activity across your storage and best practices for optimizing your storage:

- Contextual recommendations (available only in the dashboard)
- Advanced metrics (including activity metrics aggregated by bucket)
- Prefix aggregation
- Storage Lens group aggregation
- Storage Lens group aggregation
- Amazon CloudWatch publishing

Advanced metrics data is available for queries for 15 months. There are additional charges for using S3 Storage Lens with advanced metrics. For more information, see Amazon S3 pricing. For more information about free and advanced metrics, see Metrics selection.

Topics

- Viewing S3 Storage Lens metrics on the dashboards
- Viewing Amazon S3 Storage Lens metrics using a data export
- Monitor S3 Storage Lens metrics in CloudWatch
- Amazon S3 Storage Lens metrics use cases

Viewing S3 Storage Lens metrics on the dashboards

In the Amazon S3 console, S3 Storage Lens provides an interactive default dashboard that you can use to visualize insights and trends in your data. You can also use this dashboard to flag outliers and receive recommendations for optimizing storage costs and applying data-protection best practices. Your dashboard has drill-down options to generate insights at the account, bucket, AWS Region, prefix, or Storage Lens group level. If you've enabled S3 Storage Lens to work with AWS Organizations, you can also generate insights at the organization level (such as data for all accounts that are part of your AWS Organizations hierarchy). The dashboard always loads for the latest date that has metrics available.

The S3 Storage Lens default dashboard on the console is named **default-account-dashboard**. Amazon S3 pre-configures this dashboard to visualize the summarized insights and trends for your entire account and updates them daily in the S3 console. You can't modify the configuration scope of the default dashboard, but you can upgrade the metrics selection from the free metrics to the paid advanced metrics and recommendations. With advanced metrics and recommendations, you can access additional metrics and features. These features include advanced metric categories, prefix-level aggregation, contextual recommendations, and Amazon CloudWatch publishing.

You can disable the default dashboard, but you can't delete it. If you disable your default dashboard, it is no longer updated. You also will no longer receive any new daily metrics in S3 Storage Lens or in the **Account snapshot** section on the **Buckets** page. You can still see historic data in the default dashboard until the 14-day period for data queries expires. This period is 15 months if you've enabled advanced metrics and recommendations. To access this data, you can re-enable the default dashboard within the expiration period.

You can create additional S3 Storage Lens dashboards and scope them by AWS Regions, S3 buckets, or accounts. You can also scope your dashboards by organization if you've enabled Storage Lens to work with AWS Organizations. When you create or edit an S3 Storage Lens dashboard, you define your dashboard scope and metrics selection.

You can disable or delete any additional dashboards that you create.

- If you disable a dashboard, it is no longer updated, and you will no longer receive any new daily metrics. You can still see historic data for free metrics until the 14-day expiration period. If you enabled advanced metrics and recommendations for that dashboard, this period is 15 months. To access this data, you can re-enable the dashboard within the expiration period.
- If you delete your dashboard, you lose all your dashboard configuration settings. You will no longer receive any new daily metrics, and you also lose access to the historical data associated with that dashboard. If you want to access the historic data for a deleted dashboard, you must create another dashboard with the same name in the same home Region.

Topics

- Viewing an Amazon S3 Storage Lens dashboard
- Understanding your S3 Storage Lens dashboard

Viewing an Amazon S3 Storage Lens dashboard

The following procedure shows how to view an S3 Storage Lens dashboard in the S3 console. For use-case based walkthroughs that show how to use your dashboard to optimize costs, implement best practices, and improve the performance of applications that access your S3 buckets, see Amazon S3 Storage Lens metrics use cases.

> **Note**
>
> You can't use your account's root user credentials to view Amazon S3 Storage Lens dashboards. To access S3 Storage Lens dashboards, you must grant the required AWS Identity and Access Management (IAM) permissions to a new or existing IAM user. Then, sign in with those user credentials to access S3 Storage Lens dashboards. For more information, see Setting Amazon S3 Storage Lens permissions and Security best practices in IAM in the *IAM User Guide*.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.

Your dashboard opens in S3 Storage Lens. The **Snapshot for *date*** section shows the latest date that S3 Storage Lens has collected metrics for. Your dashboard always loads the latest date that has metrics available.

4. (Optional) To change the date for your S3 Storage Lens dashboard, in the date selector, choose a new date.
5. (Optional) To apply temporary filters to further limit the scope of your dashboard data, do the following:

   a. Expand the **Filters** section.

   b. To filter by specific accounts, AWS Regions, storage classes, buckets, prefixes, or Storage Lens groups, choose the options to filter by.

   > **Note**
   >
   > The **Prefixes** filter and the **Storage Lens groups** filter can't be applied at the same time.

   c. To update a filter, choose **Apply**.

   d. To remove a filter, choose the **X** next to the filter.
6. In any section in your S3 Storage Lens dashboard, to see data for a specific metric, for **Metric**, choose the metric name.
7. In any chart or visualization in your S3 Storage Lens dashboard, you can drill down into deeper levels of aggregation by using the **Accounts**, **AWS Regions**, **Storage classes**, **Buckets**, **Prefixes**, or **Storage Lens groups** tabs. For an example, see Uncover cold Amazon S3 buckets.

Understanding your S3 Storage Lens dashboard

Your S3 Storage Lens dashboard has a primary **Overview** tab, and up to five additional tabs that represent each aggregation level:

- **Accounts**
- **AWS Regions**
- **Storage classes**
- **Buckets**
- **Prefixes**

- **Storage Lens groups**

On the **Overview** tab, your dashboard data is aggregated into three different sections: **Snapshot for *date***, **Trends and distributions**, and **Top N overview**.

For more information about your S3 Storage Lens dashboard, see the following sections.

Snapshot

The **Snapshot for *date*** section shows summary metrics that S3 Storage Lens has aggregated for the date selected. These summary metrics include the following metrics:

- **Total storage** – The total amount of storage used in bytes.
- **Object count** – The total number of objects in your AWS account.
- **Average object size** – The average object size.
- **Active buckets** – The total number of active buckets in active usage with storage > 0 bytes in your account.
- **Accounts** – The number of accounts whose storage is in scope. This value is **1** unless you are using AWS Organizations and your S3 Storage Lens has trusted access with a valid service-linked role. For more information, see Using service-linked roles for Amazon S3 Storage Lens.
- **Buckets** – The total number of buckets in your account.

Metric data

For each metric that appears in the snapshot, you can see the following data:

- **Metric name** – The name of the metric.
- **Metric category** – The category that the metric is organized into.
- **Total for *date*** – The total count for the date selected.
- **% change** – The percentage change from the last snapshot date.
- **30-day trend** – A trend-line showing the changes for the metric over a 30-day period.
- **Recommendation** – A contextual recommendation based on the data that's provided in the snapshot. Recommendations are available with advanced metrics and recommendations. For more information, see Recommendations.

Metrics categories

You can optionally update your dashboard **Snapshot for *date*** section to display metrics for other categories. If you want to see snapshot data for additional metrics, you can choose from the following **Metrics categories**:

- **Cost optimization**
- **Data protection**
- **Activity** (available with advanced metrics)
- **Access management**
- **Performance**
- **Events**

The **Snapshot for *date*** section displays only a selection of metrics for each category. To see all metrics for a specific category, choose the metric in the **Trends and distributions** or **Top N overview** sections. For more information about metric categories, see Metrics categories. For a complete list of S3 Storage Lens metrics, see Amazon S3 Storage Lens metrics glossary.

Trends and distributions

The second section of the **Overview** tab is **Trends and distributions**. In the **Trends and distributions** section, you can choose two metrics to compare over a date range that you define. The **Trends and distributions** section shows the relationship between two metrics over time. This section displays charts that you can use to see the **Storage class** and **Region** distribution between the two trends that you are tracking. You can optionally drill down into a data point in one of the charts for deeper analysis.

For a walkthrough that uses the **Trends and distributions** section, see Identify buckets that don't use server-side encryption with AWS KMS for default encryption (SSE-KMS).

Top N overview

The third section of the S3 Storage Lens dashboard is **Top N overview** (sorted in ascending or descending order). This section displays your selected metrics across the top number of accounts, AWS Regions, buckets, prefixes, or Storage Lens groups. If you enabled S3 Storage Lens to work with AWS Organizations, you can also see your selected metrics across your organization.

For a walkthrough that uses the **Top N overview** section, see Identify your largest S3 buckets.

Drill down and analyze by options

To provide a fluid experience for analysis, the S3 Storage Lens dashboard provides an action menu, which appears when you choose any chart value. To use this menu, choose any chart value to see the associated metrics values, and then choose from two options in the box that appears:

- The **Drill down** action applies the selected value as a filter across all tabs of your dashboard. You can then drill down into that value for deeper analysis.
- The **Analyze by** action takes you to the **Dimension** tab that you select and applies that tab value as a filter. These tabs include **Accounts**, **AWS Regions**, **Storage classes**, **Buckets**, **Prefixes** (for dashboards that have **Advanced metrics** and **Prefix aggregation** enabled), and **Storage Lens groups** (for dashboards that have **Advanced metrics** and **Storage Lens group aggregation** enabled). With **Analyze by**, you can view the data in the context of the new dimension for deeper analysis.

The **Drill down** and **Analyze by** actions might be disabled if the outcome would yield illogical results or would not have any value. Both the **Drill down** and **Analyze by** actions apply filters on top of any existing filters across all tabs of the dashboard. You can also remove the filters as needed.

Tabs

The dimension-level tabs provide a detailed view of all values within a particular dimension. For example, the **AWS Regions** tab shows metrics for all AWS Regions, and the **Buckets** tab shows metrics for all buckets. Each dimension tab contains an identical layout consisting of four sections:

- A trend chart that displays your top *N* items within the dimension over the last 30 days for the selected metric. By default, this chart displays the top 10 items, but you can decrease it to at least 3 items or increase it up to 50 items.
- A histogram chart that shows a vertical bar chart for the selected date and metric. If you have a large number of items to display in this chart, you might need to scroll horizontally.
- A bubble analysis chart that plots all items within the dimension. This chart represents the first metric on the x axis and the second metric on the y axis. The third metric is represented by the size of the bubble.
- A metric grid view that contains each item in the dimension listed in rows. The columns represent each available metric, arranged in metrics category tabs for easier navigation.

Viewing Amazon S3 Storage Lens metrics using a data export

Amazon S3 Storage Lens metrics are generated daily in CSV or Apache Parquet-formatted metrics export files and placed in an S3 general purpose bucket in your account. From there, you can ingest the metrics export into the analytics tools of your choice, such as AWS Quick and Amazon Athena, where you can analyze storage usage and activity trends. You can also send daily metric exports to an AWS-managed S3 table bucket for immediate querying, using AWS analytics services or third-party tools.

Topics

- Using an AWS KMS key to encrypt your metrics exports
- What is an S3 Storage Lens export manifest?
- Understanding the Amazon S3 Storage Lens export schemas

Using an AWS KMS key to encrypt your metrics exports

To grant Amazon S3 Storage Lens permission to encrypt your metrics exports by using a customer managed key, you must use a key policy. To update your key policy so that you can use a KMS key to encrypt your S3 Storage Lens metrics exports, follow these steps.

To grant S3 Storage Lens permissions to encrypt data by using your KMS key

1. Sign into the AWS Management Console by using the AWS account that owns the customer managed key.
2. Open the AWS KMS console at https://console.aws.amazon.com/kms.
3. To change the AWS Region, use the **Region selector**.
4. In the left navigation pane, choose **Customer managed keys**.
5. Under **Customer managed keys**, choose the key that you want to use to encrypt the metrics exports. AWS KMS keys are Region-specific and must be in the same Region as the metrics export destination S3 bucket.
6. Under **Key policy**, choose **Switch to policy view**.
7. To update the key policy, choose **Edit**.
8. Under **Edit key policy**, add the following key policy to the existing key policy. To use this policy, replace the *user input placeholders* with your information.

```
{
```

```
    "Sid": "Allow Amazon S3 Storage Lens use of the KMS key",
     "Effect": "Allow",
    "Principal": {
        "Service": "storage-lens.s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
       "StringEquals": {
           "aws:SourceArn": "arn:aws:s3:us-east-1:       source-account-
id:storage-lens/your-dashboard-name",
           "aws:SourceAccount": "source-account-id"
        }
     }
}
```

9. Choose **Save changes**.

For more information about creating customer managed keys and using key policies, see the following topics in the *AWS Key Management Service Developer Guide*:

- Create a KMS key
- Key policies in AWS KMS

You can also use the AWS KMS PUT key policy API operation (PutKeyPolicy) to copy the key policy to the customer managed keys that you want to use to encrypt the metrics exports by using the REST API, AWS CLI, and SDKs.

What is an S3 Storage Lens export manifest?

S3 Storage Lens daily metrics exports in general-purpose buckets may be split into multiple files due to the large amount of data aggregated. The manifest file `manifest.json` describes where the metrics export files for that day are located. Whenever a new export is delivered, it's accompanied by a new manifest. Each manifest contained in the `manifest.json` file provides metadata and other basic information about the export.

The manifest information includes the following properties:

- `sourceAccountId` – The account ID of the configuration owner.
- `configId` – A unique identifier for the dashboard.
- `destinationBucket` – The destination bucket Amazon Resource Name (ARN) that the metrics export is placed in.
- `reportVersion` – The version of the export.
- `reportDate` – The date of the report.
- `reportFormat` – The format of the report.
- `reportSchema` – The schema of the report.
- `reportFiles` – The actual list of the export report files that are in the destination bucket.

Manifest destination path example:

```
user-defined-prefix/StorageLens/111122223333/example-dashboard-configuration-id/V_1/
manifests/dt=2025-03-18/manifest.json
```

The following example shows a `manifest.json` file for a CSV-formatted Storage Lens default metrics report:

```
{
   "sourceAccountId": "111122223333",
   "configId": "example-dashboard-configuration-id",
   "destinationBucket": "arn:aws:s3:::amzn-s3-demo-destination-bucket",
   "reportVersion": "V_1",
   "reportDate": "2025-07-15",
   "reportFormat": "CSV",
   "reportSchema":
 "version_number,configuration_id,report_date,aws_account_number,aws_region,storage_class,recor

   "reportFiles": [
        {
            "key": "DestinationPrefix/StorageLens/111122223333/example-dashboard-
configuration-id/V_1/reports/dt=2025-07-15/12345678-1234-1234-1234-123456789012.csv",
            "size": 1603959,
            "md5Checksum": "2177e775870def72b8d84febe1ad3574"
        }
   ]
}
```

The following example shows a `manifest.json` file for a CSV-formatted Storage Lens expanded prefixes metrics report:

```
{
   "sourceAccountId": "111122223333",
   "configId": "example-dashboard-configuration-id",
   "destinationBucket": "arn:aws:s3:::amzn-s3-demo-destination-bucket",
   "reportVersion": "V_1",
   "reportDate": "2025-11-03",
   "reportFormat": "CSV",
   "reportSchema":
 "version_number,configuration_id,report_date,aws_account_number,aws_region,storage_class,recor

   "reportFiles": [
        {
            "key": "DestinationPrefix/StorageLensExpandedPrefixes/111122223333/example-
dashboard-configuration-id/V_1/reports/dt=2025-11-03/EXAMPLE1234-56ab-78cd-90ef-
EXAMPLE11111.csv",
            "size": 1603959,
            "md5Checksum": "2177e775870def72b8d84febe1ad3574"
        }
      ]
}
```

The following example shows a `manifest.json` file for a Parquet-formatted Storage Lens default metrics report:

```
{
   "sourceAccountId": "111122223333",
   "configId": "example-dashboard-configuration-id",
   "destinationBucket": "arn:aws:s3:::amzn-s3-demo-destination-bucket",
   "reportVersion": "V_1",
   "reportDate": "2025-11-03",
   "reportFormat": "Parquet",
   "reportSchema": "message s3.storage.lens { required string version_number;
 required string configuration_id; required string report_date; required string
 aws_account_number; required string aws_region; required string storage_class;
 required string record_type; required string record_value; required string
 bucket_name; required string metric_name; required long metric_value; }",
   "reportFiles": [
      {
         "key": "DestinationPrefix/StorageLens/111122223333/example-dashboard-
configuration-id/V_1/reports/dt=2025-11-03/bd23de7c-b46a-4cf4-bcc5-b21aac5be0f5.par",
```

```
         "size": 14714,
         "md5Checksum": "b5c741ee0251cd99b90b3e8eff50b944"
      }
   ]
}
```

The following example shows a `manifest.json` file for a Parquet-formatted Storage Lens expanded prefixes metrics report:

```
{
   "sourceAccountId": "111122223333",
   "configId": "example-dashboard-configuration-id",
   "destinationBucket": "arn:aws:s3:::amzn-s3-demo-destination-bucket",
   "reportVersion": "V_1",
   "reportDate": "2025-11-03",
   "reportFormat": "Parquet",
   "reportSchema": "message s3.storage.lens { required string version_number;
 required string configuration_id; required string report_date; required string
 aws_account_number; required string aws_region; required string storage_class;
 required string record_type; required string record_value; required string
 bucket_name; required string metric_name; required long metric_value; }",
   "reportFiles": [
      {
         "key": "DestinationPrefix/StorageLensExpandedPrefixes/111122223333/example-
dashboard-configuration-id/V_1/reports/dt=2025-11-03/bd23de7c-b46a-4cf4-bcc5-
b21aac5be0f5.par",
         "size": 14714,
         "md5Checksum": "b5c741ee0251cd99b90b3e8eff50b944"
      }
   ]
}
```

You can configure your metrics export to be generated as part of your dashboard configuration in the Amazon S3 console or by using the Amazon S3 REST API, AWS CLI, and SDKs.

Understanding the Amazon S3 Storage Lens export schemas

S3 Storage Lens export schemas vary depending on your export destination. Choose the appropriate schema based on whether you're exporting to S3 general purpose buckets or S3 tables.

Topics

- Export schema for S3 general purpose buckets

- Export schemas for S3 tables

Export schema for S3 general purpose buckets

The following table contains the schema of your S3 Storage Lens metrics export when exporting to S3 general purpose buckets.

Attribute name	Data type	Column name	Description
VersionNumber	String	version_number	The version of the export report.
ConfigurationId	String	configuration_id	The configuration_id of your S3 Storage Lens configuration.
ReportDate	String	report_date	The date that the metrics were tracked.
AwsAccountNumber	String	aws_account_number	Your AWS account number.
AwsRegion	String	aws_region	The AWS Region for which the metrics are being tracked.
StorageClass	String	storage_class	The storage class of the bucket in question.
RecordType	ENUM	record_type	The type of artifact that is being reported (ACCOUNT, BUCKET, or PREFIX).
RecordValue	String	record_value	The value of the RecordType artifact.

Attribute name	Data type	Column name	Description
			Note The `record_value` is URL-encoded.
`BucketName`	String	`bucket_name`	The name of the bucket that is being reported.
`MetricName`	String	`metric_name`	The name of the metric that is being reported.
`MetricValue`	Long	`metric_value`	The value of the metric that is being reported.

Example of an S3 Storage Lens metrics export

The following is an example of an S3 Storage Lens metrics export based on this schema.

Note

You can identify metrics for Storage Lens groups by looking for the `STORAGE_LENS_GROUP_BUCKET` or `STORAGE_LENS_GROUP_ACCOUNT` values in the `record_type` column. The `record_value` column will display the Amazon Resource Name (ARN) for the Storage Lens group, for example, `arn:aws:s3:us-east-1:123456789012:storage-lens-group/slg-1`.

version_r	configuration_id	report_date	aws_account_number	aws_region	storage_class	record_type	record_value	bucket_name	metric_name	metric_value
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			StorageBytes	2478830621
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectCount	1598962
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ReplicatedStorageBytes	20000
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ReplicatedObjectCount	20
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			EncryptedStorageBytes	2478828742
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			EncryptedObjectCount	1598961
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			DeleteMarkerObjectCount	1500
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectLockEnabledStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			ObjectLockEnabledObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			CurrentVersionStorageBytes	2478830621
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			CurrentVersionObjectCount	1598962
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			NonCurrentVersionStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			NonCurrentVersionObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			IncompleteMultipartUploadStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	eu-west-1	STANDARD	ACCOUNT			IncompleteMultipartUploadObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	StorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ReplicatedStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ReplicatedObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	EncryptedStorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	EncryptedObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	DeleteMarkerObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ObjectLockEnabledStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	ObjectLockEnabledObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	CurrentVersionStorageBytes	29996800
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	CurrentVersionObjectCount	12264
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	NonCurrentVersionStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	NonCurrentVersionObjectCount	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	IncompleteMultipartUploadStorageBytes	0
V_1	sample-cmh-exclude	11/3/2020	546264889236	us-west-1	STANDARD	PREFIX	AWSLogs%2F546264889236%2FCloudTrail%2Fus	cloudtrail-log-sf	IncompleteMultipartUploadObjectCount	0

The following is an example of an S3 Storage Lens metrics export with Storage Lens groups data.

version_numbe	configuration_id	report_date	aws_account_num	aws_region	storage_class	record_type	record_value	bucket_name	metric_name	metric_value
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		StorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		EncryptedStorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		EncryptedObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		DeleteMarkerObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectLockEnabledStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ObjectLockEnabledObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		CurrentVersionStorageBytes	3128548856
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		CurrentVersionObjectCount	4440
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		NonCurrentVersionStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		NonCurrentVersionObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMultipartUploadStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMultipartUploadObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		DeleteMarkerStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedStorageBytesSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		ReplicatedObjectCountSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMPUStorageBytesOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_ACCOUNT	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180		IncompleteMPUObjectCountOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	StorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	EncryptedStorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	EncryptedObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	DeleteMarkerObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectLockEnabledStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ObjectLockEnabledObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	CurrentVersionStorageBytes	676863200
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	CurrentVersionObjectCount	3000
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	NonCurrentVersionStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	NonCurrentVersionObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMultipartUploadStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMultipartUploadObjectCount	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	DeleteMarkerStorageBytes	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedStorageBytesSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	ReplicatedObjectCountSource	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMPUStorageBytesOlderThan7Days	0
V_1	sample-sfo-exclude	11/15/23	546264889236	us-west-1	STANDARD	STORAGE_LENS_GROUP_BUCKET	arn:aws:s3:us-west-1:546264889236:storage-lens-group/age-more-than-180	cloud-trail-log-sfo	IncompleteMPUObjectCountOlderThan7Days	0

Export schemas for S3 tables

When exporting S3 Storage Lens metrics to S3 tables, the data is organized into three separate table schemas: storage metrics, bucket property metrics, and activity metrics.

Topics

- Storage metrics table schema
- Bucket property metrics table schema
- Activity metrics table schema

Storage metrics table schema

Name	Type	Description
version_number	string	Version identifier of the schema of the table
configuration_id	string	S3 Storage Lens configuration name
report_time	timestamptz	Date the S3 Storage Lens report refers to
aws_account_id	string	Account id the entry refers to
aws_region	string	Region
storage_class	string	Storage Class
record_type	string	Type of record, related to what is the level of aggregation of data. Values: ACCOUNT, BUCKET, PREFIX, LENS_GROUP.
record_value	string	Disambiguator for record types that have more than one record under them. It is used to reference the prefix
bucket_name	string	Bucket name
object_count	long	Number of objects stored for the current referenced item
storage_bytes	DECIMAL(38,0)	Number of bytes stored for the current referenced item
bucket_key_sse_kms_object_count	long	Number of objects encrypted with a customer managed key stored for the current referenced item
bucket_key_sse_kms_storage_bytes	DECIMAL(38,0)	Number of bytes encrypted with a customer managed key stored for the current referenced item

Name	Type	Description
current_version_object_count	long	Number of current version objects stored for the current referenced item
current_version_storage_bytes	DECIMAL(38,0)	Number of current version bytes stored for the current referenced item
delete_marker_object_count	long	Number of delete marker objects stored for the current referenced item
delete_marker_storage_bytes	DECIMAL(38,0)	Number of delete marker bytes stored for the current referenced item
encrypted_object_count	long	Number of encrypted objects stored for the current referenced item
encrypted_storage_bytes	DECIMAL(38,0)	Number of encrypted bytes stored for the current referenced item
incomplete_mpu_object_older_than_7_days_count	long	Number of incomplete multipart upload objects older than 7 days stored for the current referenced item
incomplete_mpu_storage_older_than_7_days_bytes	DECIMAL(38,0)	Number of incomplete multipart upload bytes stored older than 7 days for the current referenced item

Name	Type	Description
incomplete_mpu_object_count	long	Number of incomplete multipart upload objects stored for the current referenced item
incomplete_mpu_storage_bytes	DECIMAL(38,0)	Number of incomplete multipart upload bytes stored for the current referenced item
non_current_version_object_count	long	Number of non-current version objects stored for the current referenced item
non_current_version_storage_bytes	DECIMAL(38,0)	Number of non-current version bytes stored for the current referenced item
object_lock_enabled_object_count	long	Number of objects stored for for objects with lock enabled in the current referenced item
object_lock_enabled_storage_bytes	DECIMAL(38,0)	Number of bytes stored for objects with lock enabled in the current referenced item
replicated_object_count	long	Number of objects replicated for the current referenced item
replicated_storage_bytes	DECIMAL(38,0)	Number of bytes replicated for the current referenced item
replicated_object_source_count	long	Number of objects replicated as source stored for the current referenced item

Name	Type	Description
replicated_storage_source_bytes	DECIMAL(38,0)	Number of bytes replicated as source for the current referenced item
sse_kms_object_count	long	Number of objects encrypted with SSE key stored for the current referenced item
sse_kms_storage_bytes	DECIMAL(38,0)	Number of bytes encrypted with SSE key stored for the current referenced item
object_0kb_count	long	Number of objects with size equal to 0KB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_0kb_to_128kb_count	long	Number of objects with size greater than 0KB and less than or equal to 128KB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_128kb_to_256kb_count	long	Number of objects with size greater than 128KB and less than or equal to 256KB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_256kb_to_512kb_count	long	Number of objects with size greater than 256KB and less than or equal to 512KB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_512kb_to_1mb_count	long	Number of objects with size greater than 512KB and less than or equal to 1MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers

Name	Type	Description
object_1mb_to_2mb_count	long	Number of objects with size greater than 1MB and less than or equal to 2MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_2mb_to_4mb_count	long	Number of objects with size greater than 2MB and less than or equal to 4MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_4mb_to_8mb_count	long	Number of objects with size greater than 4MB and less than or equal to 8MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_8mb_to_16mb_count	long	Number of objects with size greater than 8MB and less than or equal to 16MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_16mb_to_32mb_count	long	Number of objects with size greater than 16MB and less than or equal to 32MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_32mb_to_64mb_count	long	Number of objects with size greater than 32MB and less than or equal to 64MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_64mb_to_128mb_count	long	Number of objects with size greater than 64MB and less than or equal to 128MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers

Name	Type	Description
object_128mb_to_256mb_count	long	Number of objects with size greater than 128MB and less than or equal to 256MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_256mb_to_512mb_count	long	Number of objects with size greater than 256MB and less than or equal to 512MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_512mb_to_1gb_count	long	Number of objects with size greater than 512MB and less than or equal to 1GB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_1gb_to_2gb_count	long	Number of objects with size greater than 1GB and less than or equal to 2GB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_2gb_to_4gb_count	long	Number of objects with size greater than 2GB and less than or equal to 4GB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_larger_than_4gb_count	long	Number of objects with size greater than 4GB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers

Bucket property metrics table schema

Name	Type	Description
version_number	string	Version identifier of the schema of the table

Name	Type	Description
configuration_id	string	S3 Storage Lens configuration name
report_time	timestamptz	Date the S3 Storage Lens report refers to
aws_account_id	string	Account id the entry refers to
record_type	string	Type of record, related to what is the level of aggregation of data. Values: ACCOUNT, BUCKET, PREFIX, LENS GROUP.
record_value	string	Disambiguator for record types that have more than one record under them. It is used to reference the prefix.
aws_region	string	Region
storage_class	string	Storage Class
bucket_name	string	Bucket name
versioning_enabled_bucket_count	long	Number of buckets with versioning enabled for the current referenced item
mfa_delete_enabled_bucket_count	long	Number of buckets with MFA delete enabled for the current referenced item
sse_kms_enabled_bucket_count	long	Number of buckets with KMS enabled for the current referenced item

Name	Type	Description
object_ownership_bucket_owner_enforced_bucket_count	long	Number of buckets with Object Ownership bucket owner enforced for the current referenced item
object_ownership_bucket_owner_preferred_bucket_count	long	Number of buckets with Object Ownership bucket owner preferred for the current referenced item
object_ownership_object_writer_bucket_count	long	Number of buckets with Object Ownership object writer for the current referenced item
transfer_acceleration_enabled_bucket_count	long	Number of buckets with transfer acceleration enabled for the current referenced item
event_notification_enabled_bucket_count	long	Number of buckets with event notification enabled for the current referenced item
transition_lifecycle_rule_count	long	Number of transition lifecycle rules for the current referenced item

Name	Type	Description
expiration_lifecycle_rule_count	long	Number of expiration lifecycle rules for the current referenced item
non_current_version_transition_lifecycle_rule_count	long	Number of noncurrent version transition lifecycle rules for the current referenced item
non_current_version_expiration_lifecycle_rule_count	long	Number of noncurrent version expiration lifecycle rules for the current referenced item
abort_incomplete_multipart_upload_lifecycle_rule_count	long	Number of abort incomplete multipart upload lifecycle rules for the current referenced item
expired_object_delete_marker_lifecycle_rule_count	long	Number of expire object delete marker lifecycle rules for the current referenced item
same_region_replication_rule_count	long	Number of Same-Region Replication rule count for the current referenced item

Name	Type	Description
cross_region_replication_rule_count	long	Number of Cross-Region Replication rule count for the current referenced item
same_account_replication_rule_count	long	Number of Same-account replication rule count for the current referenced item
cross_account_replication_rule_count	long	Number of Cross-account replication rule count for the current referenced item
invalid_destination_replication_rule_count	long	Number of buckets with Invalid destination replication for the current referenced item

Activity metrics table schema

Name	Type	Description
version_number	string	Version identifier of the schema of the table
configuration_id	string	S3 Storage Lens configuration name
report_time	timestamp tz	Date the S3 Storage Lens report refers to
aws_account_id	string	Account id the entry refers to

Name	Type	Description
aws_region	string	Region
storage_class	string	Storage Class
record_type	string	Type of record, related to what is the level of aggregation of data. Values: ACCOUNT, BUCKET, PREFIX.
record_value	string	Disambiguator for record types that have more than one record under them. It is used to reference the prefix
bucket_name	string	Bucket name
all_request_count	long	Number of _all_ requests for the current referenced item
all_sse_kms_encrypted_request_count	long	Number of KMS encrypted requests for the current referenced item
all_unsupported_sig_request_count	long	Number of unsupported sig requests for the current referenced item
all_unsupported_tls_request_count	long	Number of unsupported TLS requests for the current referenced item
bad_request_error_400_count	long	Number of 400 bad request errors for the current referenced item
delete_request_count	long	Number of delete requests for the current referenced item

Name	Type	Description
downloaded_bytes	decimal(0,0)	Number of downloaded bytes for the current referenced item
error_4xx_count	long	Number of 4xx errors for the current referenced item
error_5xx_count	long	Number of 5xx errors for the current referenced item
forbidden_error_403_count	long	Number of 403 forbidden errors for the current referenced item
get_request_count	long	Number of get requests for the current referenced item
head_request_count	long	Number of head requests for the current referenced item
internal_server_error_500_count	long	Number of 500 internal server errors for the current referenced item
list_request_count	long	Number of list requests for the current referenced item
not_found_error_404_count	long	Number of 404 not found errors for the current referenced item
ok_status_200_count	long	Number of 200 OK requests for the current referenced item
partial_content_status_206_count	long	Number of 206 partial content requests for the current referenced item
post_request_count	long	Number of post requests for the current referenced item

Name	Type	Description
put_request_count	long	Number of put requests for the current referenced item
select_request_count	long	Number of select requests for the current referenced item
select_returned_bytes	decimal(0,0)	Number of bytes returned by select requests for the current referenced item
select_scanned_bytes	decimal(0,0)	Number of bytes scanned by select requests for the current referenced item
service_unavailable_error_503_count	long	Number of 503 service unavailable errors for the current referenced item
uploaded_bytes	decimal(0,0)	Number of uploaded bytes for the current referenced item
average_first_byte_latency	long	Average per-request time between when an S3 bucket receives a complete request and when it starts returning the response, measured over the past 24 hours
average_total_request_latency	long	Average elapsed per-request time between the first byte received and the last byte sent to an S3 bucket, measured over the past 24 hours
read_0kb_request_count	long	Number of GetObject requests with data sizes of 0KB, including both range-based requests and whole object requests
read_0kb_to_128kb_request_count	long	Number of GetObject requests with data sizes greater than 0KB and up to 128KB, including both range-based requests and whole object requests

Name	Type	Description
read_128kb_to_256kb_request_count	long	Number of GetObject requests with data sizes greater than 128KB and up to 256KB, including both range-based requests and whole object requests
read_256kb_to_512kb_request_count	long	Number of GetObject requests with data sizes greater than 256KB and up to 512KB, including both range-based requests and whole object requests
read_512kb_to_1mb_request_count	long	Number of GetObject requests with data sizes greater than 512KB and up to 1MB, including both range-based requests and whole object requests
read_1mb_to_2mb_request_count	long	Number of GetObject requests with data sizes greater than 1MB and up to 2MB, including both range-based requests and whole object requests
read_2mb_to_4mb_request_count	long	Number of GetObject requests with data sizes greater than 2MB and up to 4MB, including both range-based requests and whole object requests
read_4mb_to_8mb_request_count	long	Number of GetObject requests with data sizes greater than 4MB and up to 8MB, including both range-based requests and whole object requests
read_8mb_to_16mb_request_count	long	Number of GetObject requests with data sizes greater than 8MB and up to 16MB, including both range-based requests and whole object requests
read_16mb_to_32mb_request_count	long	Number of GetObject requests with data sizes greater than 16MB and up to 32MB, including both range-based requests and whole object requests

Name	Type	Description
read_32mb_to_64mb_request_count	long	Number of GetObject requests with data sizes greater than 32MB and up to 64MB, including both range-based requests and whole object requests
read_64mb_to_128mb_request_count	long	Number of GetObject requests with data sizes greater than 64MB and up to 128MB, including both range-based requests and whole object requests
read_128mb_to_256mb_request_count	long	Number of GetObject requests with data sizes greater than 128MB and up to 256MB, including both range-based requests and whole object requests
read_256mb_to_512mb_request_count	long	Number of GetObject requests with data sizes greater than 256MB and up to 512MB, including both range-based requests and whole object requests
read_512mb_to_1gb_request_count	long	Number of GetObject requests with data sizes greater than 512MB and up to 1GB, including both range-based requests and whole object requests
read_1gb_to_2gb_request_count	long	Number of GetObject requests with data sizes greater than 1GB and up to 2GB, including both range-based requests and whole object requests
read_2gb_to_4gb_request_count	long	Number of GetObject requests with data sizes greater than 2GB and up to 4GB, including both range-based requests and whole object requests
read_larger_than_4gb_request_count	long	Number of GetObject requests with data sizes greater than 4GB, including both range-based requests and whole object requests

Name	Type	Description
write_0kb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes of 0KB
write_0kb_to_128kb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 0KB and up to 128KB
write_128kb_to_256kb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 128KB and up to 256KB
write_256kb_to_512kb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 256KB and up to 512KB
write_512kb_to_1mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 512KB and up to 1MB
write_1mb_to_2mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 1MB and up to 2MB
write_2mb_to_4mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 2MB and up to 4MB
write_4mb_to_8mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 4MB and up to 8MB
write_8mb_to_16mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 8MB and up to 16MB

Name	Type	Description
write_16mb_to_32mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 16MB and up to 32MB
write_32mb_to_64mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 32MB and up to 64MB
write_64mb_to_128mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 64MB and up to 128MB
write_128mb_to_256mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 128MB and up to 256MB
write_256mb_to_512mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 256MB and up to 512MB
write_512mb_to_1gb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 512MB and up to 1GB
write_1gb_to_2gb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 1GB and up to 2GB
write_2gb_to_4gb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 2GB and up to 4GB

Name	Type	Description
write_larger_than_4gb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 4GB
concurrent_put_503_error_count	long	Number of 503 errors that are generated due to concurrent writes to the same object
cross_region_request_count	long	Number of requests that originate from a client in different Region than bucket's home Region
cross_region_transferred_bytes	decimal(0,0)	Number of bytes that are transferred from calls in different Region than bucket's home Region
cross_region_without_replication_request_count	long	Number of requests that originate from a client in different Region than bucket's home Region, excluding cross-region replication requests
cross_region_without_replication_transferred_bytes	decimal(0,0)	Number of bytes that are transferred from calls in different Region than bucket's home Region, excluding cross-region replication bytes
inregion_request_count	long	Number of requests that originate from a client in same Region as bucket's home Region
inregion_transferred_bytes	decimal(0,0)	Number of bytes that are transferred from calls from same Region as bucket's home Region

Name	Type	Description
`unique_objects_accessed_daily_count`	long	Number of objects that were accessed at least once in last 24 hrs

Monitor S3 Storage Lens metrics in CloudWatch

You can publish S3 Storage Lens metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch features, such as alarms and triggered actions, metric math, and anomaly detection, to monitor and take action on S3 Storage Lens metrics. In addition, CloudWatch API operations enable applications, including third-party providers, to access your S3 Storage Lens metrics. For more information about CloudWatch features, see the Amazon CloudWatch User Guide.

You can enable the CloudWatch publishing option for new or existing dashboard configurations by using the Amazon S3 console, Amazon S3 REST API, AWS CLI, and AWS SDKs. Dashboards that are upgraded to S3 Storage Lens advanced metrics and recommendations can use the CloudWatch publishing option. For S3 Storage Lens advanced metrics and recommendations pricing, see Amazon S3 pricing. No additional CloudWatch metrics publishing charges apply; however, other CloudWatch charges, such as dashboards, alarms, and API calls, do apply. For more information, see Amazon CloudWatch pricing.

S3 Storage Lens metrics are published to CloudWatch in the account that owns the S3 Storage Lens configuration. After you enable the CloudWatch publishing option within advanced metrics, you can access account-level and bucket-level metrics by configuration ID, account, bucket (for bucket-level metrics only), Region, and storage class in CloudWatch. Prefix-level metrics are not available in CloudWatch.

> **Note**
>
> S3 Storage Lens metrics are daily metrics and are published to CloudWatch once per day. When you query S3 Storage Lens metrics in CloudWatch, the period for the query must be 1 day (86400 seconds). After your daily S3 Storage Lens metrics appear in your S3 Storage Lens dashboard in the Amazon S3 console, it can take a few hours for these same metrics to appear in CloudWatch. When you enable the CloudWatch publishing option for

S3 Storage Lens metrics for the first time, it can take up to 24 hours for your metrics to publish to CloudWatch.

After you enable the CloudWatch publishing option, you can use the following CloudWatch features to monitor and analyze your S3 Storage LensStorage Lens data:

- Dashboards – Use CloudWatch dashboards to create customized S3 Storage Lens dashboards. Share your CloudWatch dashboard with people who don't have direct access to your AWS account, across teams, with stakeholders, and with people external to your organizations.
- Alarms and triggered actions – Configure alarms that watch metrics and take action when a threshold is breached. For example, you can configure an alarm that sends an Amazon SNS notification when the **Incomplete Multipart Upload Bytes** metric exceeds 1 GB for three consecutive days.
- Anomaly detection – Enable anomaly detection to continuously analyze metrics, determine normal baselines, and surface anomalies. You can create an anomaly detection alarm based on the expected value of a metric. For example, you can monitor anomalies for the **Object Lock Enabled Bytes** metric to detect unauthorized removal of Object Lock settings.
- Metric math – You can also use metric math to query multiple S3 Storage Lens metrics and use math expressions to create new time series based on these metrics. For example, you can create a new metric to get the average object size by dividing `StorageBytes` by `ObjectCount`.

For more information about the CloudWatch publishing option for S3 Storage Lens metrics, see the following topics.

Topics

- S3 Storage Lens metrics and dimensions
- Enabling CloudWatch publishing for S3 Storage Lens
- Working with S3 Storage Lens metrics in CloudWatch

S3 Storage Lens metrics and dimensions

To send S3 Storage Lens metrics to CloudWatch, you must enable the CloudWatch publishing option within S3 Storage Lens advanced metrics. After advanced metrics are enabled, you can use CloudWatch dashboards to monitor S3 Storage Lens metrics alongside other application metrics and create a unified view of your operational health. You can use dimensions to filter your S3

Storage Lens metrics in CloudWatch by organization, account, bucket, storage class, Region, and metrics configuration ID.

S3 Storage Lens metrics are published to CloudWatch in the account that owns the S3 Storage Lens configuration. After you enable the CloudWatch publishing option within advanced metrics, you can access account-level and bucket-level metrics by configuration ID, account, bucket (for bucket-level metrics only), Region, and storage class in CloudWatch. Prefix-level metrics are not available in CloudWatch.

> **Note**
>
> S3 Storage Lens metrics are daily metrics and are published to CloudWatch once per day. When you query S3 Storage Lens metrics in CloudWatch, the period for the query must be 1 day (86400 seconds). After your daily S3 Storage Lens metrics appear in your S3 Storage Lens dashboard in the Amazon S3 console, it can take a few hours for these same metrics to appear in CloudWatch. When you enable the CloudWatch publishing option for S3 Storage Lens metrics for the first time, it can take up to 24 hours for your metrics to publish to CloudWatch.

For more information about S3 Storage Lens metrics and dimensions in CloudWatch, see the following topics.

Topics

- Metrics
- Dimensions

Metrics

S3 Storage Lens metrics are available as metrics within CloudWatch. S3 Storage Lens metrics are published to the `AWS/S3/Storage-Lens` namespace. This namespace is only for S3 Storage Lens metrics. Amazon S3 bucket, request, and replication metrics are published to the `AWS/S3` namespace.

S3 Storage Lens metrics are published to CloudWatch in the account that owns the S3 Storage Lens configuration. After you enable the CloudWatch publishing option within advanced metrics, you can access account-level and bucket-level metrics by configuration ID, account, bucket (for

bucket-level metrics only), Region, and storage class in CloudWatch. Prefix-level metrics are not available in CloudWatch.

In S3 Storage Lens, metrics are aggregated and stored only in the designated home Region. S3 Storage Lens metrics are also published to CloudWatch in the home Region that you specify in the S3 Storage Lens configuration.

For a complete list of S3 Storage Lens metrics, including a list of those metrics available in CloudWatch, see Amazon S3 Storage Lens metrics glossary.

> **Note**
>
> The valid statistic for S3 Storage Lens metrics in CloudWatch is Average. For more information about statistics in CloudWatch, see CloudWatch statistics definitions in the *Amazon CloudWatch User Guide*.

Granularity of S3 Storage Lens metrics in CloudWatch

S3 Storage Lens offers metrics at organization, account, bucket, and prefix granularity. S3 Storage Lens publishes organization, account, and bucket-level S3 Storage Lens metrics to CloudWatch. Prefix-level S3 Storage Lens metrics are not available in CloudWatch.

For more information about the granularity of S3 Storage Lens metrics available in CloudWatch, see the following list:

- **Organization** – Metrics aggregated across the member accounts in your organization. S3 Storage Lens publishes metrics for member accounts to CloudWatch in the management account.
  - **Organization and account** – Metrics for the member accounts in your organization.
  - **Organization and bucket** – Metrics for Amazon S3 buckets in the member accounts of your organization.
- **Account** (Non-organization level) – Metrics aggregated across the buckets in your account.
- **Bucket** (Non-organization level) – Metrics for a specific bucket. In CloudWatch, S3 Storage Lens publishes these metrics to the AWS account that created the S3 Storage Lens configuration. S3 Storage Lens publishes these metrics only for non-organization configurations.

Dimensions

When S3 Storage Lens sends data to CloudWatch, dimensions are attached to each metric. Dimensions are categories that describe the characteristics of metrics. You can use dimensions to filter the results that CloudWatch returns.

For example, all S3 Storage Lens metrics in CloudWatch have the `configuration_id` dimension. You can use this dimension to differentiate between metrics associated with a specific S3 Storage Lens configuration. The `organization_id` identifies organization-level metrics. For more information about dimensions in CloudWatch, see Dimensions in the *CloudWatch User Guide*.

Different dimensions are available for S3 Storage Lens metrics depending on the granularity of the metrics. For example, you can use the `organization_id` dimension to filter organization-level metrics by the AWS Organizations ID. However, you can't use this dimension for bucket and account-level metrics. For more information, see Filtering metrics using dimensions.

To see which dimensions are available for your S3 Storage Lens configuration, see the following table.

Dimension	Description	Bucket	Account	Organization	Organization and account	Organization and bucket
`configuration_id`	The dashboard name for the S3 Storage Lens configuration reported in the metrics	✓	✓	✓	✓	✓
`metrics_version`	The version of the S3 Storage Lens metrics. The metrics version has a fixed value of `1.0`.	✓	✓	✓	✓	✓
`organization_id`	The AWS Organizations ID for the metrics	✗	✗	✓	✓	✓
`aws_account_number`	The AWS account that's associated with the metrics	✓	✓	✗	✓	✓
`aws_region`	The AWS Region for the metrics	✓	✓	✓	✓	✓

Dimension	Description	Bucket	Account	Organization	Organization and account	Organization and bucket
bucket_name	The name of the S3 bucket that's reported in the metrics	✓	✗	✗	✓	✗
storage_class	The storage class for the bucket that's reported in the metrics	✓	✓	✓	✓	✓
record_type	The granularity of the metrics: ORGANIZATION, ACCOUNT, BUCKET	✓ BUCKET	✓ ACCOUNT	✓ ORGANIZATION	✓ ACCOUNT	✓ BUCKET

Enabling CloudWatch publishing for S3 Storage Lens

You can publish S3 Storage Lens metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch features, such as alarms and triggered actions, metric math, and anomaly detection, to monitor and take action on S3 Storage Lens metrics. In addition, CloudWatch API operations enable applications, including third-party providers, to access your S3 Storage Lens metrics. For more information about CloudWatch features, see the Amazon CloudWatch User Guide.

S3 Storage Lens metrics are published to CloudWatch in the account that owns the S3 Storage Lens configuration. After you enable the CloudWatch publishing option within advanced metrics, you can access account-level and bucket-level metrics by configuration ID, account, bucket (for bucket-level metrics only), Region, and storage class in CloudWatch. Prefix-level metrics are not available in CloudWatch.

You can enable CloudWatch support for new or existing dashboard configurations by using the S3 console, Amazon S3 REST APIs, AWS CLI, and AWS SDKs. The CloudWatch publishing option is available for dashboards that are upgraded to S3 Storage Lens advanced metrics and recommendations. For S3 Storage Lens advanced metrics and recommendations pricing, see Amazon S3 pricing. No additional CloudWatch metrics publishing charges apply; however, other CloudWatch charges, such as dashboards, alarms, and API calls, do apply.

To enable the CloudWatch publishing option for S3 Storage Lens metrics, see the following topics.

> **Note**
>
> S3 Storage Lens metrics are daily metrics and are published to CloudWatch once per day. When you query S3 Storage Lens metrics in CloudWatch, the period for the query must be 1 day (86400 seconds). After your daily S3 Storage Lens metrics appear in your S3 Storage Lens dashboard in the Amazon S3 console, it can take a few hours for these same metrics to appear in CloudWatch. When you enable the CloudWatch publishing option for S3 Storage Lens metrics for the first time, it can take up to 24 hours for your metrics to publish to CloudWatch.
> Currently, S3 Storage Lens metrics cannot be consumed through CloudWatch streams.

Using the S3 console

When you update an S3 Storage Lens dashboard, you can't change the dashboard name or home Region. You also can't change the scope of the default dashboard, which is scoped to your entire account's storage.

To update an S3 Storage Lens dashboard to enable CloudWatch publishing

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **S3 Storage Lens**, **Dashboards**.
3. Choose the dashboard that you want to edit, and then choose **Edit.**
4. Under **Metrics selection**, choose **Advanced metrics and recommendations**.

   Advanced metrics and recommendations are available for an additional charge. Advanced metrics and recommendations include a 15-month period for data queries, usage metrics aggregated at the prefix level, activity metrics aggregated by bucket, the CloudWatch publishing option, and contextual recommendations that help you optimize storage costs and apply data-protection best practices. For more information, see Amazon S3 pricing.
5. Under **Select Advanced metrics and recommendations features**, select **CloudWatch publishing**.

> **Important**
>
> If your configuration enables prefix aggregation for usage metrics, prefix-level metrics will not be published to CloudWatch. Only bucket, account, and organization-level S3 Storage Lens metrics are published to CloudWatch.

6. Choose **Save changes**.

To create a new S3 Storage Lens dashboard that enables CloudWatch support

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. Choose **Create dashboard**.
4. Under **General**, define the following configuration options:

   a. For **Dashboard name**, enter your dashboard name.

   Dashboard names must be fewer than 65 characters and must not contain special characters or spaces. You can't change the dashboard name after you create your dashboard.

   b. Choose the **Home Region** for your dashboard.

   Metrics for all Regions included in this dashboard scope are stored centrally in the designated home Region. In CloudWatch, S3 Storage Lens metrics are also available in the home Region. You can't change the home Region after you create your dashboard.

5. (Optional) To add tags, choose **Add tag** and enter the tag **Key** and **Value**.

> **Note**
>
> You can add up to 50 tags to your dashboard configuration.

6. Define the scope for your configuration:

   a. If you're creating an organization-level configuration, choose the accounts to include in the configuration: **Include all accounts in your configuration** or **Limit the scope to your signed-in account**.

> **Note**
>
> When you create an organization-level configuration that includes all accounts, you can include or exclude only Regions, not buckets.

b. Choose the Regions and buckets that you want S3 Storage Lens to include in the dashboard configuration by doing the following:

- To include all Regions, choose **Include Regions and buckets**.
- To include specific Regions, clear **Include all Regions**. Under **Choose Regions to include**, choose the Regions that you want S3 Storage Lens to include in the dashboard.
- To include specific buckets, clear **Include all buckets**. Under **Choose buckets to include**, choose the buckets that you want S3 Storage Lens to include in the dashboard.

> **Note**
>
> You can choose up to 50 buckets.

7. For **Metrics selection**, choose **Advanced metrics and recommendations**.

   For more information about advanced metrics and recommendations pricing, see Amazon S3 pricing.

8. Under **Advanced metrics and recommendations features**, select the options that you want to enable:

- **Advanced metrics**
- **CloudWatch publishing**

> **Important**
>
> If you enable prefix aggregation for your S3 Storage Lens configuration, prefix-level metrics will not be published to CloudWatch. Only bucket, account, and organization-level S3 Storage Lens metrics are published to CloudWatch.

- **Prefix aggregation**

> **Note**
>
> For more information about advanced metrics and recommendations features, see Metrics selection.

9. If you enabled **Advanced metrics**, select the **Advanced metrics categories** that you want to display in your S3 Storage Lens dashboard:

   - **Activity metrics**
   - **Detailed status code metrics**
   - **Advanced cost optimization metrics**
   - **Advanced data protection metrics**

   For more information about metrics categories, see Metrics categories. For a complete list of metrics, see Amazon S3 Storage Lens metrics glossary.

10. (Optional) Configure your metrics export.

    For more information about how to configure a metrics export, see step Using the S3 console.

11. Choose **Create dashboard**.

Using the AWS CLI

The following AWS CLI example enables the CloudWatch publishing option by using a S3 Storage Lens organization-level advanced metrics and recommendations configuration. To use this example, replace the *`user input placeholders`* with your own information.

```
aws s3control put-storage-lens-configuration --account-id=555555555555 --config-
id=your-configuration-id --region=us-east-1 --storage-lens-configuration=file://./
config.json

config.json
{
  "Id": "SampleS3StorageLensConfiguration",
  "AwsOrg": {
    "Arn": "arn:aws:organizations::123456789012:organization/o-abcdefgh"
  },
  "AccountLevel": {
```

```
    "ActivityMetrics": {
      "IsEnabled":true
    },
    "AdvancedCostOptimizationMetrics": {
      "IsEnabled":true
    },
    "AdvancedDataProtectionMetrics": {
      "IsEnabled":true
    },
    "DetailedStatusCodesMetrics": {
      "IsEnabled":true
    },
    "BucketLevel": {
      "ActivityMetrics": {
        "IsEnabled":true
      },
      "AdvancedCostOptimizationMetrics": {
        "IsEnabled":true
      },
      "DetailedStatusCodesMetrics": {
        "IsEnabled":true
      },
      "PrefixLevel":{
        "StorageMetrics":{
          "IsEnabled":true,
          "SelectionCriteria":{
            "MaxDepth":5,
            "MinStorageBytesPercentage":1.25,
            "Delimiter":"/"
          }
        }
      }
    }
  },
  "Exclude": {
    "Regions": [
      "eu-west-1"
    ],
    "Buckets": [
      "arn:aws:s3:::amzn-s3-demo-source-bucket "
    ]
  },
  "IsEnabled": true,
  "DataExport": {
```

```
    "S3BucketDestination": {
      "OutputSchemaVersion": "V_1",
      "Format": "CSV",
      "AccountId": "111122223333",
      "Arn": "arn:aws:s3:::amzn-s3-demo-destination-bucket",
      "Prefix": "prefix-for-your-export-destination",
      "Encryption": {
        "SSES3": {}
      }
    },
    "CloudWatchMetrics": {
      "IsEnabled": true
    }
  }
}
```

Using the AWS SDK for Java

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.ActivityMetrics;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.CloudWatchMetrics;
import com.amazonaws.services.s3control.model.Format;
import com.amazonaws.services.s3control.model.Include;
import com.amazonaws.services.s3control.model.OutputSchemaVersion;
import com.amazonaws.services.s3control.model.PrefixLevel;
import com.amazonaws.services.s3control.model.PrefixLevelStorageMetrics;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.S3BucketDestination;
import com.amazonaws.services.s3control.model.SSES3;
import com.amazonaws.services.s3control.model.SelectionCriteria;
import com.amazonaws.services.s3control.model.StorageLensAwsOrg;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensDataExport;
import com.amazonaws.services.s3control.model.StorageLensDataExportEncryption;
import com.amazonaws.services.s3control.model.StorageLensTag;
```

```
import java.util.Arrays;
import java.util.List;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateAndUpdateDashboard {

    public static void main(String[] args) {
        String configurationId = "ConfigurationId";
        String sourceAccountId = "Source Account ID";
        String exportAccountId = "Destination Account ID";
        String exportBucketArn = "arn:aws:s3:::amzn-s3-demo-destination-bucket"; //
 The destination bucket for your metrics export must be in the same Region as your S3
 Storage Lens configuration.
        String awsOrgARN = "arn:aws:organizations::123456789012:organization/o-
abcdefgh";
        Format exportFormat = Format.CSV;

        try {
            SelectionCriteria selectionCriteria = new SelectionCriteria()
                    .withDelimiter("/")
                    .withMaxDepth(5)
                    .withMinStorageBytesPercentage(10.0);
            PrefixLevelStorageMetrics prefixStorageMetrics = new
 PrefixLevelStorageMetrics()
                    .withIsEnabled(true)
                    .withSelectionCriteria(selectionCriteria);
            BucketLevel bucketLevel = new BucketLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
 AdvancedDataProtectionMetrics().withIsEnabled(true))
                    .withDetailedStatusCodesMetrics(new
 DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withPrefixLevel(new
 PrefixLevel().withStorageMetrics(prefixStorageMetrics));
            AccountLevel accountLevel = new AccountLevel()
                    .withActivityMetrics(new ActivityMetrics().withIsEnabled(true))
                    .withAdvancedCostOptimizationMetrics(new
 AdvancedCostOptimizationMetrics().withIsEnabled(true))
                    .withAdvancedDataProtectionMetrics(new
 AdvancedDataProtectionMetrics().withIsEnabled(true))
```

```
                    .withDetailedStatusCodesMetrics(new
DetailedStatusCodesMetrics().withIsEnabled(true))
                    .withBucketLevel(bucketLevel);

            Include include = new Include()
                    .withBuckets(Arrays.asList("arn:aws:s3:::amzn-s3-demo-bucket"))
                    .withRegions(Arrays.asList("us-west-2"));

            StorageLensDataExportEncryption exportEncryption = new
StorageLensDataExportEncryption()
                    .withSSES3(new SSES3());
            S3BucketDestination s3BucketDestination = new S3BucketDestination()
                    .withAccountId(exportAccountId)
                    .withArn(exportBucketArn)
                    .withEncryption(exportEncryption)
                    .withFormat(exportFormat)
                    .withOutputSchemaVersion(OutputSchemaVersion.V_1)
                    .withPrefix("Prefix");
            CloudWatchMetrics cloudWatchMetrics = new CloudWatchMetrics()
                    .withIsEnabled(true);
            StorageLensDataExport dataExport = new StorageLensDataExport()
                    .withCloudWatchMetrics(cloudWatchMetrics)
                    .withS3BucketDestination(s3BucketDestination);

            StorageLensAwsOrg awsOrg = new StorageLensAwsOrg()
                    .withArn(awsOrgARN);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withInclude(include)
                    .withDataExport(dataExport)
                    .withAwsOrg(awsOrg)
                    .withIsEnabled(true);

            List<StorageLensTag> tags = Arrays.asList(
                    new StorageLensTag().withKey("key-1").withValue("value-1"),
                    new StorageLensTag().withKey("key-2").withValue("value-2")
            );

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();
```

```
            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
                    .withTags(tags)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Using the REST API

To enable the CloudWatch publishing option by using the Amazon S3 REST API, you can use PutStorageLensConfiguration.

Next steps

After you enable the CloudWatch publishing option, you can access your S3 Storage Lens metrics in CloudWatch. You also can leverage CloudWatch features to monitor and analyze your S3 Storage Lens data in CloudWatch. For more information, see the following topics:

- S3 Storage Lens metrics and dimensions
- Working with S3 Storage Lens metrics in CloudWatch

Working with S3 Storage Lens metrics in CloudWatch

You can publish S3 Storage Lens metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch features, such as alarms and triggered actions, metric math, and anomaly detection, to monitor and take action on S3 Storage Lens metrics. In addition, CloudWatch API operations enable applications,

including third-party providers, to access your S3 Storage Lens metrics. For more information about CloudWatch features, see the Amazon CloudWatch User Guide.

You can enable the CloudWatch publishing option for new or existing dashboard configurations by using the Amazon S3 console, Amazon S3 REST APIs, AWS CLI, and AWS SDKs. The CloudWatch publishing option is available for dashboards that are upgraded to S3 Storage Lens advanced metrics and recommendations. For S3 Storage Lens advanced metrics and recommendations pricing, see Amazon S3 pricing. No additional CloudWatch metrics publishing charges apply; however, other CloudWatch charges, such as dashboards, alarms, and API calls, do apply. For more information, see Amazon CloudWatch pricing.

S3 Storage Lens metrics are published to CloudWatch in the account that owns the S3 Storage Lens configuration. After you enable the CloudWatch publishing option within advanced metrics, you can access account-level and bucket-level metrics by configuration ID, account, bucket (for bucket-level metrics only), Region, and storage class in CloudWatch. Prefix-level metrics are not available in CloudWatch.

Note

S3 Storage Lens metrics are daily metrics and are published to CloudWatch once per day. When you query S3 Storage Lens metrics in CloudWatch, the period for the query must be 1 day (86400 seconds). After your daily S3 Storage Lens metrics appear in your S3 Storage Lens dashboard in the Amazon S3 console, it can take a few hours for these same metrics to appear in CloudWatch. When you enable the CloudWatch publishing option for S3 Storage Lens metrics for the first time, it can take up to 24 hours for your metrics to publish to CloudWatch.
Currently, S3 Storage Lens metrics cannot be consumed through CloudWatch streams.

For more information about working with S3 Storage Lens metrics in CloudWatch, see the following topics.

Topics

- Working with CloudWatch dashboards
- Setting alarms, triggering actions, and using anomaly detection
- Filtering metrics using dimensions
- Calculating new metrics with metric math

- Using search expressions in graphs

Working with CloudWatch dashboards

You can use CloudWatch dashboards to monitor S3 Storage Lens metrics alongside other application metrics and create a unified view of your operational health. Dashboards are customizable home pages in the CloudWatch console that you can use to monitor your resources in a single view.

CloudWatch has broad permissions control that doesn't support limiting access to a specific set of metrics or dimensions. Users in your account or organization who have access to CloudWatch will have access to metrics for all S3 Storage Lens configurations where the CloudWatch support option is enabled. You can't manage permissions for specific dashboards as you can in S3 Storage Lens. For more information about CloudWatch permissions, see Managing access permissions to your CloudWatch resources in the *Amazon CloudWatch User Guide*.

For more information about using CloudWatch dashboards and configuring permissions, see Using Amazon CloudWatch dashboards and Sharing CloudWatch dashboards in the *Amazon CloudWatch User Guide*.

Setting alarms, triggering actions, and using anomaly detection

You can configure CloudWatch alarms that watch S3 Storage Lens metrics in CloudWatch and take action when a threshold is breached. For example, you can configure an alarm that sends an Amazon SNS notification when the **Incomplete Multipart Upload Bytes** metric exceeds 1 GB for three consecutive days.

You can also enable anomaly detection to continuously analyze your S3 Storage Lens metrics, determine normal baselines, and surface anomalies. You can create an anomaly detection alarm based on a metric's expected value. For example, you can monitor anomalies for the **Object Lock Enabled Bytes** metric to detect unauthorized removal of Object Lock settings.

For more information and examples, see Using Amazon CloudWatch alarms and Creating an alarm from a metric on a graph in the *Amazon CloudWatch User Guide*.

Filtering metrics using dimensions

You can use dimensions to filter S3 Storage Lens metrics in the CloudWatch console. For example, you can filter by `configuration_id`, `aws_account_number`, `aws_region`, `bucket_name`, and more.

S3 Storage Lens supports multiple dashboard configurations per account. This means that different configurations can include the same bucket. When these metrics are published to CloudWatch, the bucket will have duplicate metrics within CloudWatch. To view metrics only for a specific S3 Storage Lens configuration in CloudWatch, you can use the `configuration_id` dimension. When you filter by `configuration_id`, you see only the metrics that are associated with the configuration that you identify.

For more information about filtering by configuration ID, see Searching for available metrics in the *Amazon CloudWatch User Guide*.

Calculating new metrics with metric math

You can use metric math to query multiple S3 Storage Lens metrics and use math expressions to create new time series based on these metrics. For example, you can create a new metric for unencrypted objects by subtracting Encrypted Objects from Object Count. You can also create a metric to get the average object size by dividing `StorageBytes` by `ObjectCount` or the number bytes accessed on one day by dividing `BytesDownloaded` by `StorageBytes`.

For more information, see Using metric math in the *Amazon CloudWatch User Guide*.

Using search expressions in graphs

With S3 Storage Lens metrics, you can create a search expression. For example, you can create a search expression for all metrics that are named **IncompleteMultipartUploadStorageBytes** and add SUM to the expression. With this search expression, you can see your total incomplete multipart upload bytes across all dimensions of your storage in a single metric.

This example shows the syntax that you would use to create a search expression for all metrics named **IncompleteMultipartUploadStorageBytes**.

```
SUM(SEARCH('{AWS/S3/Storage-
Lens,aws_account_number,aws_region,configuration_id,metrics_version,record_type,storage_class}
 MetricName="IncompleteMultipartUploadStorageBytes"', 'Average',86400))
```

For more information about this syntax, see CloudWatch search expression syntax in the *Amazon CloudWatch User Guide*. To create a CloudWatch graph with a search expression, see Creating a CloudWatch graph with a search expressionin the *Amazon CloudWatch User Guide*.

Amazon S3 Storage Lens metrics use cases

You can use your Amazon S3 Storage Lens dashboard to visualize insights and trends, flag outliers, and receive recommendations. S3 Storage Lens metrics are organized into categories that align with key use cases. You can use these metrics to do the following:

- Identify cost-optimization opportunities
- Apply data-protection best practices
- Apply access-management best practices
- Improve the performance of application workloads

For example, with cost-optimization metrics, you can identify opportunities to reduce your Amazon S3 storage costs. You can identify buckets with multipart uploads that are more than 7-days old or buckets that are accumulating noncurrent versions.

Similarly, you can use data-protection metrics to identify buckets that aren't following data-protection best practices within your organization. For example, you can identify buckets that don’t use AWS Key Management Service keys (SSE-KMS) for default encryption or don't have S3 Versioning enabled.

With S3 Storage Lens access-management metrics, you can identify bucket settings for S3 Object Ownership so that you can migrate access control list (ACL) permissions to bucket policies and disable ACLs.

If you have S3 Storage Lens advanced metrics enabled, you can use detailed status-code metrics to get counts for successful or failed requests that you can use to troubleshoot access or performance issues.

With advanced metrics, you can also access additional cost-optimization and data-protection metrics that you can use to identify opportunities to further reduce your overall S3 storage costs and better align with best practices for protecting your data. For example, advanced cost-optimization metrics include lifecycle rule counts that you can use to identify buckets that don't have lifecycle rules to expire incomplete multipart uploads that are more than 7 days old. Advanced data-protection metrics include replication rule counts.

For more information about metrics categories, see Metrics categories. For a complete list of S3 Storage Lens metrics, see Amazon S3 Storage Lens metrics glossary.

Topics

- Using Amazon S3 Storage Lens to optimize your storage costs
- Using S3 Storage Lens to protect your data
- Using S3 Storage Lens to audit Object Ownership settings
- Using S3 Storage Lens metrics to improve performance

Using Amazon S3 Storage Lens to optimize your storage costs

You can use S3 Storage Lens cost-optimization metrics to reduce the overall cost of your S3 storage. Cost-optimization metrics can help you confirm that you've configured Amazon S3 cost effectively and according to best practices. For example, you can identify the following cost-optimization opportunities:

- Buckets with incomplete multipart uploads older than 7 days
- Buckets that are accumulating numerous noncurrent versions
- Buckets that don't have lifecycle rules to abort incomplete multipart uploads
- Buckets that don't have lifecycle rules to expire noncurrent versions objects
- Buckets that don't have lifecycle rules to transition objects to a different storage class

You can then use this data to add additional lifecycle rules to your buckets.

The following examples show how you can use cost- optimization metrics in your S3 Storage Lens dashboard to optimize your storage costs.

Topics

- Identify your largest S3 buckets
- Uncover cold Amazon S3 buckets
- Locate incomplete multipart uploads
- Reduce the number of noncurrent versions retained
- Identify buckets that don't have lifecycle rules and review lifecycle rule counts

Identify your largest S3 buckets

You pay for storing objects in S3 buckets. The rate that you're charged depends on your objects' sizes, how long you store the objects, and their storage classes. With S3 Storage Lens, you get a centralized view of all the buckets in your account. To see all the buckets in all of your

organization's accounts, you can configure an AWS Organizations-level S3 Storage Lens dashboard. From this dashboard view, you can identify your largest buckets.

Step 1: Identify your largest buckets

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.

   When the dashboard opens, you can see the latest date that S3 Storage Lens has collected metrics for. Your dashboard always loads to the latest date that has metrics available.
4. To see a ranking of your largest buckets by the **Total storage** metric for a selected date range, scroll down to the **Top N overview for *date*** section.

   You can toggle the sort order to show the smallest buckets. You can also adjust the **Metric** selection to rank your buckets by any of the available metrics. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the trend. This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.

   > **Note**
   >
   > With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.
5. For more detailed insights about your buckets, scroll up to the top of the page, and then choose the **Bucket** tab.

   On the **Bucket** tab, you can see details such as the recent growth rate, the average object size, the largest prefixes, and the number of objects.

Step 2: Navigate to your buckets and investigate

After you've identified your largest S3 buckets, you can navigate to each bucket within the S3 console to view the objects in the bucket, understand its associated workload, and identify its internal owners. You can contact the bucket owners to find out whether the growth is expected or whether the growth needs further monitoring and control.

Uncover cold Amazon S3 buckets

If you have S3 Storage Lens advanced metrics enabled, you can use activity metrics to understand how cold your S3 buckets are. A "cold" bucket is one whose storage is no longer accessed (or very rarely accessed). This lack of activity typically indicates that the bucket's objects aren't frequently accessed.

Activity metrics, such as **GET Requests** and **Download Bytes**, indicate how often your buckets are accessed each day. To understand the consistency of the access pattern and to spot buckets that are no longer being accessed at all, you can trend this data over several months. The **Retrieval rate** metric, which is computed as **Download bytes / Total storage**, indicates the proportion of storage in a bucket that is accessed daily.

> **Note**
>
> Download bytes are duplicated in cases where the same object is downloaded multiple times during the day.

Prerequisite

To see activity metrics in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations** and then select **Activity metrics**. For more information, see Using the S3 console.

Step 1: Identify active buckets

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. Choose the **Bucket** tab, and then scroll down to the **Bubble analysis by buckets for *date*** section.

   In the **Bubble analysis by buckets for *date*** section, you can plot your buckets on multiple dimensions by using any three metrics to represent the **X-axis**, **Y-axis**, and **Size** of the bubble.
5. To find buckets that have gone cold, for **X-axis**, **Y-axis**, and **Size**, choose the **Total storage**, **% retrieval rate**, and **Average object size** metrics.

6. In the **Bubble analysis by buckets for *date*** section, locate any buckets with retrieval rates of zero (or near zero) and a larger relative storage size, and choose the bubble that represents the bucket.

   A box will appear with choices for more granular insights. Do one of the following:

   a. To update the **Bucket** tab to display metrics only for the selected bucket, choose **Drill down**, and then choose **Apply**.
   b. To aggregate your bucket-level data to by account, AWS Region, storage class, or bucket, choose **Analyze by** and then make a choice for **Dimension**. For example, to aggregate by storage class, choose **Storage class** for **Dimension**.

   To find buckets that have gone cold, do a bubble analysis using the **Total storage**, **% retrieval rate**, and **Average object size** metrics. Look for any buckets with retrieval rates of zero (or near zero) and a larger relative storage size.

   The **Bucket** tab of your dashboard updates to display data for your selected aggregation or filter. If you aggregated by storage class or another dimension, that new tab opens in your dashboard (for example, the **Storage class** tab).

Step 2: Investigate cold buckets

From here, you can identify the owners of cold buckets in your account or organization and find out if that storage is still needed. You can then optimize costs by configuring lifecycle expiration configurations for these buckets or archiving the data in one of the Amazon Glacier storage classes.

To avoid the problem of cold buckets going forward, you can automatically transition your data by using S3 Lifecycle configurations for your buckets, or you can enable auto-archiving with S3 Intelligent-Tiering.

You can also use step 1 to identify hot buckets. Then, you can ensure that these buckets use the correct S3 storage class to ensure that they serve their requests most effectively in terms of performance and cost.

Locate incomplete multipart uploads

You can use multipart uploads to upload very large objects (up to 50 TB) as a set of parts for improved throughput and quicker recovery from network issues. In cases where the multipart upload process doesn't finish, the incomplete parts remain in the bucket (in an unusable state).

These incomplete parts incur storage costs until the upload process is finished, or until the incomplete parts are removed. For more information, see Uploading and copying objects using multipart upload in Amazon S3.

With S3 Storage Lens, you can identify the number of incomplete multipart upload bytes in your account or across your entire organization, including incomplete multipart uploads that are more than 7 days old. For a complete list of incomplete multipart upload metrics, see Amazon S3 Storage Lens metrics glossary.

As a best practice, we recommend configuring lifecycle rules to expire incomplete multipart uploads that are older than a specific number of days. When you create your lifecycle rule to expire incomplete multipart uploads, we recommend 7 days as a good starting point.

Step 1: Review overall trends for incomplete multipart uploads

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. In the **Snapshot for *date*** section, under **Metrics categories**, choose **Cost optimization**.

   The **Snapshot for *date*** section updates to display **Cost optimization** metrics, which include **Incomplete multipart upload bytes greater than 7 days old**.

   In any chart in your S3 Storage Lens dashboard, you can see metrics for incomplete multipart uploads. You can use these metrics to further assess the impact of incomplete multipart upload bytes on your storage, including their contribution to overall growth trends. You can also drill down to deeper levels of aggregation, using the **Account**, **AWS Region**, **Bucket**, or **Storage class** tabs for a deeper analysis of your data. For an example, see Uncover cold Amazon S3 buckets.

Step 2: Identify buckets that have the most incomplete multipart upload bytes but don't have lifecycle rules to abort incomplete multipart uploads

Prerequisite

To see the **Abort incomplete multipart upload lifecycle rule count** metric in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations**, and then select **Advanced cost optimization metrics**. For more information, see Using the S3 console.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. To identify specific buckets that are accumulating incomplete multipart uploads greater than 7 days old, go to the **Top N overview for *date*** section.

   By default, the **Top N overview for *date*** section displays metrics for the top 3 buckets. You can increase or decrease the number of buckets in the **Top N** field. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the trend. (This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.)

   > **Note**
   >
   > With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.

5. For **Metric**, choose **Incomplete multipart upload bytes greater than 7 days old** in the **Cost optimization** category.

   Under **Top *number* buckets**, you can see the buckets with the most incomplete multipart upload storage bytes that are greater than 7 days old.
6. To view more detailed bucket-level metrics for incomplete multipart uploads, choose the **Bucket** tab.
7. Scroll down to the **Buckets** section. For **Metrics categories**, select **Cost optimization**. Then clear **Summary**.

   The **Buckets** list updates to display all the available **Cost optimization** metrics for the buckets shown.
8. To filter the **Buckets** list to display only specific cost-optimization metrics, choose the preferences icon ().
9. Clear the toggles for all cost-optimization metrics until only **Incomplete multipart upload bytes greater than 7 days old** and **Abort incomplete multipart upload lifecycle rule count** remain selected.

10. (Optional) Under **Page size**, choose the number of buckets to display in the list.
11. Choose **Confirm**.

    The **Buckets** list updates to display bucket-level metrics for incomplete multipart uploads and lifecycle rule counts. You can use this data to identify buckets that have the most incomplete multipart upload bytes that are greater than 7 days old and are missing lifecycle rules to abort incomplete multipart uploads. Then, you can navigate to these buckets in the S3 console and add lifecycle rules to delete abandoned incomplete multipart uploads.

Step 3: Add a lifecycle rule to delete incomplete multipart uploads after 7 days

To automatically manage incomplete multipart uploads, you can use the S3 console to create a lifecycle configuration to expire incomplete multipart upload bytes from a bucket after a specified number of days. For more information, see Configuring a bucket lifecycle configuration to delete incomplete multipart uploads.

Reduce the number of noncurrent versions retained

When enabled, S3 Versioning retains multiple distinct copies of the same object that you can use to quickly recover data if an object is accidentally deleted or overwritten. If you've enabled S3 Versioning without configuring lifecycle rules to transition or expire noncurrent versions, a large number of previous noncurrent versions can accumulate, which can have storage-cost implications. For more information, see Retaining multiple versions of objects with S3 Versioning.

Step 1: Identify buckets with the most noncurrent object versions

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. In the **Snapshot for *date*** section, under **Metric categories**, choose **Cost optimization**.

   The **Snapshot for *date*** section updates to display **Cost optimization** metrics, which include the metric for **% noncurrent version bytes**. The **% noncurrent version bytes** metric represents the proportion of your total storage bytes that is attributed to noncurrent versions, within the dashboard scope and for the selected date.

> **Note**
>
> If your **% noncurrent version bytes** is greater than 10 percent of your storage at the account level, you might be storing too many object versions.

5. To identify specific buckets that are accumulating a large number of noncurrent versions:

   a. Scroll down to the **Top N overview for *date*** section. For **Top N**, enter the number of buckets that you would like to see data for.

   b. For **Metric**, choose **% noncurrent version bytes**.

      Under **Top *number* buckets**, you can see the buckets (for the number that you specified) with the highest **% noncurrent version bytes**. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the trend. This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.

      > **Note**
      >
      > With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.

   c. To view more detailed bucket-level metrics for noncurrent object versions, choose the **Bucket** tab.

      In any chart or visualization in your S3 Storage Lens dashboard, you can drill down to deeper levels of aggregation, using the **Account**, **AWS Region**, **Storage class**, or **Bucket** tabs. For an example, see Uncover cold Amazon S3 buckets.

   d. In the **Buckets** section, for **Metric categories**, select **Cost optimization**. Then, clear **Summary**.

      You can now see the **% noncurrent version bytes** metric, along with other metrics related to noncurrent versions.

Step 2: Identify buckets that are missing transition and expiration lifecycle rules for managing noncurrent versions

Prerequisite

To see the **Noncurrent version transition lifecycle rule count** and **Noncurrent version expiration lifecycle rule count** metrics in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations**, and then select **Advanced cost optimization metrics**. For more information, see Using the S3 console.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. In your Storage Lens dashboard, choose the **Bucket** tab.
5. Scroll down to the **Buckets** section. For **Metrics categories**, select **Cost optimization**. Then clear **Summary**.

   The **Buckets** list updates to display all the available **Cost optimization** metrics for the buckets shown.
6. To filter the **Buckets** list to display only specific cost-optimization metrics, choose the preferences icon
   ().
7. Clear the toggles for all cost-optimization metrics until only the following remain selected:
   - **% noncurrent version bytes**
   - **Noncurrent version transition lifecycle rule count**
   - **Noncurrent version expiration lifecycle rule count**
8. (Optional) Under **Page size**, choose the number of buckets to display in the list.
9. Choose **Confirm**.

   The **Buckets** list updates to display metrics for noncurrent version bytes and noncurrent version lifecycle rule counts. You can use this data to identify buckets that have a high percentage of noncurrent version bytes but are missing transition and expiration lifecycle rules. Then, you can navigate to these buckets in the S3 console and add lifecycle rules to these buckets.

Step 3: Add lifecycle rules to transition or expire noncurrent object versions

After you've determined which buckets require further investigation, you can navigate to the buckets within the S3 console and add a lifecycle rule to expire noncurrent versions after a specified number of days. Alternatively, to reduce costs while still retaining noncurrent versions, you can configure a lifecycle rule to transition noncurrent versions to one of the Amazon Glacier storage classes. For more information, see Specifying a lifecycle rule for a versioning-enabled bucket.

Identify buckets that don't have lifecycle rules and review lifecycle rule counts

S3 Storage Lens provides S3 Lifecycle rule count metrics that you can use to identify buckets that are missing lifecycle rules. To find buckets that don't have lifecycle rules, you can use the **Total buckets without lifecycle rules** metric. A bucket with no S3 Lifecycle configuration might have storage that you no longer need or can migrate to a lower-cost storage class. You can also use lifecycle rule count metrics to identify buckets that are missing specific types of lifecycle rules, such as expiration or transition rules.

Prerequisite

To see lifecycle rule count metrics and the **Total buckets without lifecycle rules** metric in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations**, and then select **Advanced cost optimization metrics**. For more information, see Using the S3 console.

Step 1: Identify buckets without lifecycle rules

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. To identify specific buckets without lifecycle rules, scroll down to the **Top N overview for *date*** section.

   By default, the **Top N overview for *date*** section displays metrics for the top 3 buckets. In the **Top N** field, you can increase the number of buckets. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the trend. This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.

> **Note**
>
> With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.

5. For **Metric**, choose **Total buckets without lifecycle rules** from the **Cost optimization** category.
6. Review the following data for **Total buckets without lifecycle rules**:

   - **Top *number* accounts** - See which accounts that have the most buckets without lifecycle rules.
   - **Top *number* Regions** - View a breakdown of buckets without lifecycle rules by Region.
   - **Top *number* buckets** - See which buckets don't have lifecycle rules.

   In any chart or visualization in your S3 Storage Lens dashboard, you can drill down to deeper levels of aggregation, using the **Account**, **AWS Region**, **Storage class**, or **Bucket** tabs. For an example, see Uncover cold Amazon S3 buckets.

   After you identify which buckets don't have lifecycle rules, you can also review specific lifecycle rule counts for your buckets.

Step 2: Review lifecycle rule counts for your buckets

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the dashboard that you want to view.
4. In your S3 Storage Lens dashboard, choose the **Bucket** tab.
5. Scroll down to the **Buckets** section. Under **Metrics categories**, select **Cost optimization**. Then clear **Summary**.

   The **Buckets** list updates to display all the available **Cost optimization** metrics for the buckets shown.
6. To filter the **Buckets** list to display only specific cost-optimization metrics, choose the preferences icon ().

7. Clear the toggles for all cost-optimization metrics until only the following remain selected:

   - **Transition lifecycle rule count**
   - **Expiration lifecycle rule count**
   - **Noncurrent version transition lifecycle rule count**
   - **Noncurrent version expiration lifecycle rule count**
   - **Abort incomplete multipart upload lifecycle rule count**
   - **Total lifecycle rule count**
8. (Optional) Under **Page size**, choose the number of buckets to display in the list.
9. Choose **Confirm**.

   The **Buckets** list updates to display lifecycle rule count metrics for your buckets. You can use this data to identify buckets without lifecycle rules or buckets that are missing specific kinds of lifecycle rules, for example, expiration or transition rules. Then, you can navigate to these buckets in the S3 console and add lifecycle rules to these buckets.

Step 3: Add lifecycle rules

After you've identified buckets with no lifecycle rules, you can add lifecycle rules. For more information, see Setting an S3 Lifecycle configuration on a bucket and Examples of S3 Lifecycle configurations.

Using S3 Storage Lens to protect your data

You can use Amazon S3 Storage Lens data-protection metrics to identify buckets where data-protection best practices haven't been applied. You can use these metrics to take action and apply standard settings that align with best practices for protecting your data across the buckets in your account or organization. For example, you can use data-protection metrics to identify buckets that don't use AWS Key Management Service (AWS KMS) keys (SSE-KMS) for default encryption or requests that use AWS Signature Version 2 (SigV2).

The following use cases provide strategies for using your S3 Storage Lens dashboard to identify outliers and apply data-protection best practices across your S3 buckets.

Topics

- Identify buckets that don't use server-side encryption with AWS KMS for default encryption (SSE-KMS)

- Identify buckets that have S3 Versioning enabled
- Identify requests that use AWS Signature Version 2 (SigV2)
- Count the total number of replication rules for each bucket
- Identify percentage of Object Lock bytes

Identify buckets that don't use server-side encryption with AWS KMS for default encryption (SSE-KMS)

With Amazon S3 default encryption, you can set the default encryption behavior for an S3 bucket. For more information, see the section called “Setting default bucket encryption”.

You can use the **SSE-KMS enabled bucket count** and **% SSE-KMS enabled buckets** metrics to identify buckets that use server-side encryption with AWS KMS keys (SSE-KMS) for default encryption. S3 Storage Lens also provides metrics for unencrypted bytes, unencrypted objects, encrypted bytes, and encrypted objects. For a complete list of metrics, see Amazon S3 Storage Lens metrics glossary.

You can analyze SSE-KMS encryption metrics in the context of general encryption metrics to identify buckets that don't use SSE-KMS. If you want to use SSE-KMS for all the buckets in your account or organization, you can then update the default encryption settings for these buckets to use SSE-KMS. In addition to SSE-KMS, you can use server-side encryption with Amazon S3 managed keys (SSE-S3) or customer-provided keys (SSE-C). For more information, see Protecting data with encryption.

Step 1: Identify which buckets are using SSE-KMS for default encryption

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In the **Trends and distributions** section, choose **% SSE-KMS enabled bucket count** for the primary metric and **% encrypted bytes** for the secondary metric.

   The **Trend for *date*** chart updates to display trends for SSE-KMS and encrypted bytes.
5. To view more granular, bucket-level insights for SSE-KMS:

   a. Choose a point on the chart. A box will appear with choices for more granular insights.

b. Choose the **Buckets** dimension. Then choose **Apply**.

6. In the **Distribution by buckets for *date*** chart, choose the **SSE-KMS enabled bucket count** metric.
7. You can now see which buckets have SSE-KMS enabled and which do not.

Step 2: Update bucket default encryption settings

Now that you've determined which buckets use SSE-KMS in the context of your **% encrypted bytes**, you can identify buckets that don't use SSE-KMS. You can then optionally navigate to these buckets within the S3 console and update their default encryption settings to use SSE-KMS or SSE-S3. For more information, see Configuring default encryption.

Identify buckets that have S3 Versioning enabled

When enabled, the S3 Versioning feature retains multiple versions of the same object that can be used to quickly recover data if an object is accidentally deleted or overwritten. You can use the **Versioning-enabled bucket count** metric to see which buckets use S3 Versioning. Then, you can take action in the S3 console to enable S3 Versioning for other buckets.

Step 1: Identify buckets that have S3 Versioning enabled

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In the **Trends and distributions** section, choose **Versioning-enabled bucket count** for the primary metric and **Buckets** for the secondary metric.

   The **Trend for *date*** chart updates to display trends for S3 Versioning enabled buckets. Right below the trends line, you can see the **Storage class distribution** and **Region distribution** subsections.
5. To view more granular insights for any of the buckets that you see in the **Trend for *date*** chart so that you can perform a deeper analysis, do the following:

   a. Choose a point on the chart. A box will appear with choices for more granular insights.

   b. Choose a dimension to apply to your data for deeper analysis: **Account**, **AWS Region**, **Storage class**, or **Bucket**. Then choose **Apply**.

6. In the **Bubble analysis by buckets for *date*** section, choose the **Versioning-enabled bucket count**, **Buckets**, and **Active buckets** metrics.

   The **Bubble analysis by buckets for *date*** section updates to display data for the metrics that you selected. You can use this data to see which buckets have S3 Versioning enabled in the context of your total bucket count. In the **Bubble analysis by buckets for *date*** section, you can plot your buckets on multiple dimensions by using any three metrics to represent the **X-axis**, **Y-axis**, and **Size** of the bubble.

Step 2: Enable S3 Versioning

After you've identified buckets that have S3 Versioning enabled, you can identify buckets that have never had S3 Versioning enabled or are versioning suspended. Then, you can optionally enable versioning for these buckets in the S3 console. For more information, see Enabling versioning on buckets.

Identify requests that use AWS Signature Version 2 (SigV2)

You can use the **All unsupported signature requests** metric to identify requests that use AWS Signature Version 2 (SigV2). This data can help you identify specific applications that are using SigV2. You can then migrate these applications to AWS Signature Version 4 (SigV4).

SigV4 is the recommended signing method for all new S3 applications. SigV4 provides improved security and is supported in all AWS Regions. For more information, see Amazon S3 update - SigV2 deprecation period extended & modified.

Prerequisite

To see **All unsupported signature requests** in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations** and then select **Advanced data protection metrics**. For more information, see Using the S3 console.

Step 1: Examine SigV2 signing trends by AWS account, Region, and bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. To identify specific buckets, accounts, and Regions with requests that use SigV2:

a. Under **Top N overview for *date***, in **Top N**, enter the number of buckets that you would like to see data for.

b. For **Metric**, choose **All unsupported signature requests** from the **Data protection** category.

The **Top N overview for *date*** updates to display data for SigV2 requests by account, AWS Region, and bucket. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the trend. This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.

> **Note**
>
> With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.

Step 2: Identify buckets that are accessed by applications through SigV2 requests

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In your Storage Lens dashboard, choose the **Bucket** tab.
5. Scroll down to the **Buckets** section. Under **Metrics categories**, choose **Data protection**. Then clear **Summary**.

   The **Buckets** list updates to display all the available **Data protection** metrics for the buckets shown.
6. To filter the **Buckets** list to display only specific data-protection metrics, choose the preferences icon ().
7. Clear the toggles for all data-protection metrics until only the following metrics remain selected:

   - **All unsupported signature requests**

- **% all unsupported signature requests**

8. (Optional) Under **Page size**, choose the number of buckets to display in the list.
9. Choose **Confirm**.

   The **Buckets** list updates to display bucket-level metrics for SigV2 requests. You can use this data to identify specific buckets that have SigV2 requests. Then, you can use this information to migrate your applications to SigV4. For more information, see Authenticating Requests (AWS Signature Version 4) in the *Amazon Simple Storage Service API Reference*.

Count the total number of replication rules for each bucket

S3 Replication enables automatic, asynchronous copying of objects across Amazon S3 buckets. Buckets that are configured for object replication can be owned by the same AWS account or by different accounts. For more information, see Replicating objects within and across Regions.

You can use S3 Storage Lens replication rule count metrics to get detailed per-bucket information about your buckets that are configured for replication. This information includes replication rules within and across buckets and Regions.

Prerequisite

To see replication rule count metrics in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations** and then select **Advanced data protection metrics**. For more information, see Using the S3 console.

Step 1: Count the total number of replication rules for each bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In your Storage Lens dashboard, choose the **Bucket** tab.
5. Scroll down to the **Buckets** section. Under **Metrics categories**, choose **Data protection**. Then clear **Summary**.
6. To filter the **Buckets** list to display only replication rule count metrics, choose the preferences icon
   ().

7. Clear the toggles for all data-protection metrics until only the replication rule count metrics remain selected:

   - **Same-Region Replication rule count**
   - **Cross-Region Replication rule count**
   - **Same-account replication rule count**
   - **Cross-account replication rule count**
   - **Total replication rule count**

8. (Optional) Under **Page size**, choose the number of buckets to display in the list.
9. Choose **Confirm**.

Step 2: Add replication rules

After you have a per-bucket replication rule count, you can optionally create additional replication rules. For more information, see Examples for configuring live replication.

Identify percentage of Object Lock bytes

With S3 Object Lock, you can store objects by using a *write-once-read-many (WORM)* model. You can use Object Lock to help prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. You can enable Object Lock only when you create a bucket and also enable S3 Versioning. However, you can edit the retention period for individual object versions or apply legal holds for buckets that have Object Lock enabled. For more information, see Locking objects with Object Lock.

You can use Object Lock metrics in S3 Storage Lens to see the **% Object Lock bytes** metric for your account or organization. You can use this information to identify buckets in your account or organization that aren't following your data-protection best practices.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In the **Snapshot** section, under **Metrics categories**, choose **Data protection**.

The **Snapshot** section updates to display data-protection metrics, including the **% Object Lock bytes** metric. You can see the overall percentage of Object Lock bytes for your account or organization.

5. To see the **% Object Lock bytes** per bucket, scroll down to the **Top N overview** section.

   To get object-level data for Object Lock, you can also use the **Object Lock object count** and **% Object Lock objects** metrics.

6. For **Metric**, choose **% Object Lock bytes** from the **Data protection** category.

   By default, the **Top N overview for *date*** section displays metrics for the top 3 buckets. In the **Top N** field, you can increase the number of buckets. The **Top N overview for *date*** section also shows the percentage change from the prior day or week and a spark-line to visualize the trend. This trend is a 14-day trend for free metrics and a 30-day trend for advanced metrics and recommendations.

   > **Note**
   >
   > With S3 Storage Lens advanced metrics and recommendations, metrics are available for queries for 15 months. For more information, see Metrics selection.

7. Review the following data for **% Object Lock bytes**:

   - **Top *number* accounts** - See which accounts have the highest and lowest **% Object Lock bytes**.
   - **Top *number* Regions** - View a breakdown of **% Object Lock bytes** by Region.
   - **Top *number* buckets** - See which buckets have the highest and lowest **% Object Lock bytes**.

Using S3 Storage Lens to audit Object Ownership settings

Amazon S3 Object Ownership is an S3 bucket-level setting that you can use to disable access control lists (ACLs) and control ownership of the objects in your bucket. If you set Object Ownership to bucket owner enforced, you can disable access control lists (ACLs) and take ownership of every object in your bucket. This approach simplifies access management for data stored in Amazon S3.

By default, when another AWS account uploads an object to your S3 bucket, that account (the object writer) owns the object, has access to it, and can grant other users access to it through ACLs. You can use Object Ownership to change this default behavior.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. Therefore, we recommend that you disable ACLs, except in circumstances where you must control access for each object individually. By setting Object Ownership to bucket owner enforced, you can disable ACLs and rely on policies for access control. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

With S3 Storage Lens access-management metrics, you can identify buckets that don't have disabled ACLs. After identifying these buckets, you can migrate ACL permissions to policies and disable ACLs for these buckets.

Topics

- Step 1: Identify general trends for Object Ownership settings
- Step 2: Identify bucket-level trends for Object Ownership settings
- Step 3: Update your Object Ownership setting to bucket owner enforced to disable ACLs

Step 1: Identify general trends for Object Ownership settings

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In the **Snapshot for *date*** section, under **Metrics categories**, choose **Access management**.

   The **Snapshot for *date*** section updates to display the **% Object Ownership bucket owner enforced** metric. You can see the overall percentage of buckets in your account or organization that use the bucket owner enforced setting for Object Ownership to disable ACLs.

Step 2: Identify bucket-level trends for Object Ownership settings

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.

3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. To view more detailed bucket-level metrics, choose the **Bucket** tab.
5. In the **Distribution by buckets for *date*** section, choose the **% Object Ownership bucket owner enforced** metric.

   The chart updates to show a per-bucket breakdown for **% Object Ownership bucket owner enforced**. You can see which buckets use the bucket owner enforced setting for Object Ownership to disable ACLs.
6. To view the bucket owner enforced settings in context, scroll down to the **Buckets** section. For **Metrics categories**, select **Access management**. Then clear **Summary**.

   The **Buckets** list displays data for all three Object Ownership settings: bucket owner enforced, bucket owner preferred, and object writer.
7. To filter the **Buckets** list to display metrics only for a specific Object Ownership setting, choose the preferences icon
   ( ).
8. Clear the metrics that you don't want to see.
9. (Optional) Under **Page size**, choose the number of buckets to display in the list.
10. Choose **Confirm**.

Step 3: Update your Object Ownership setting to bucket owner enforced to disable ACLs

After you've identified buckets that use the object writer and bucket owner preferred setting for Object Ownership, you can migrate your ACL permissions to bucket policies. When you've finished migrating your ACL permissions, you can then update your Object Ownership settings to bucket owner enforced in order to disable ACLs. For more information, see Prerequisites for disabling ACLs.

Using S3 Storage Lens metrics to improve performance

If you have S3 Storage Lens advanced metrics enabled, you can use detailed status-code metrics to get counts for successful or failed requests. You can use this information to troubleshoot access or performance issues. Detailed status-code metrics show counts for HTTP status codes, such as 403 Forbidden and 503 Service Unavailable. You can examine overall trends for detailed status-code metrics across S3 buckets, accounts, and organizations. Then, you can drill down into bucket-level metrics to identify workloads that are currently accessing these buckets and causing errors.

For example, you can look at the **403 Forbidden error count** metric to identify workloads that are accessing buckets without the correct permissions applied. After you've identified these workloads, you can do a deep dive outside of S3 Storage Lens to troubleshoot your 403 Forbidden errors.

This example shows you how to do a trend analysis for the 403 Forbidden error by using the **403 Forbidden error count** and the **% 403 Forbidden errors** metrics. You can use these metrics to identify workloads that are accessing buckets without the correct permissions applied. You can do a similar trend analysis for any of the other **Detailed status code metrics**. For more information, see Amazon S3 Storage Lens metrics glossary.

Prerequisite

To see **Detailed status code metrics** in your S3 Storage Lens dashboard, you must enable S3 Storage Lens **Advanced metrics and recommendations**, and then select **Detailed status code metrics**. For more information, see Using the S3 console.

Topics

- Step 1: Do a trend analysis for an individual HTTP status code
- Step 2: Analyze error counts by bucket
- Step 3: Troubleshoot errors

Step 1: Do a trend analysis for an individual HTTP status code

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In the **Trends and distributions** section, for **Primary metric**, choose **403 Forbidden error count** from the **Detailed status codes** category. For **Secondary metric**, choose **% 403 Forbidden errors**.
5. Scroll down to the **Top N overview for *date*** section. For **Metrics**, choose **403 Forbidden error count** or **% 403 Forbidden errors** from the **Detailed status codes** category.

   The **Top N overview for *date*** section updates to display the top 403 Forbidden error counts by account, AWS Region, and bucket.

Step 2: Analyze error counts by bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, **Dashboards**.
3. In the **Dashboards** list, choose the name of the dashboard that you want to view.
4. In your Storage Lens dashboard, choose the **Bucket** tab.
5. Scroll down to the **Buckets** section. For **Metrics categories**, select **Detailed status code** metrics. Then clear **Summary**.

   The **Buckets** list updates to display all the available detailed status code metrics. You can use this information to see which buckets have a large proportion of certain HTTP status codes and which status codes are common across buckets.
6. To filter the **Buckets** list to display only specific detailed status-code metrics, choose the preferences icon ().
7. Clear the toggles for any detailed status-code metrics that you don't want to view in the **Buckets** list.
8. (Optional) Under **Page size**, choose the number of buckets to display in the list.
9. Choose **Confirm**.

   The **Buckets** list displays error count metrics for the number of buckets that you specified. You can use this information to identify specific buckets that are experiencing many errors and troubleshoot errors by bucket.

Step 3: Troubleshoot errors

After you identify buckets with a high proportion of specific HTTP status codes, you can troubleshoot these errors. For more information, see the following:

- Why am I getting a 403 Forbidden error when I try to upload files in Amazon S3?
- Why am I getting a 403 Forbidden error when I try to modify a bucket policy in Amazon S3?
- How do I troubleshoot 403 Forbidden errors from my Amazon S3 bucket where all the resources are from the same AWS account?
- How do I troubleshoot an HTTP 500 or 503 error from Amazon S3?

Viewing metrics in AWS managed S3 Tables

Amazon S3 Storage Lens can export your storage analytics and insights to S3 Tables, enabling you to query your Storage Lens metrics using SQL with AWS analytics services like Amazon Athena, Amazon EMR, Amazon SageMaker Studio (SMStudio), and other AWS analytics tools. When you configure S3 Storage Lens to export to S3 Tables, your metrics are automatically stored in read-only Apache Iceberg tables in the AWS-managed `aws-s3` table bucket.

This integration provides structured data access for querying Storage Lens metrics using standard SQL, analytics integration with AWS analytics services, historical analysis capabilities, and cost optimization with no additional charges for exporting to AWS-managed S3 Tables.

Topics

- Exporting S3 Storage Lens metrics to S3 Tables
- Table naming for S3 Storage Lens export to S3 Tables
- Understanding S3 Storage Lens table schemas
- Permissions for S3 Storage Lens tables
- Querying S3 Storage Lens data with analytics tools
- Using AI assistants with S3 Storage Lens tables

Exporting S3 Storage Lens metrics to S3 Tables

You can configure Amazon S3 Storage Lens to export your storage analytics and insights to S3 Tables. When you enable S3 Tables export, your metrics are automatically stored in read-only Apache Iceberg tables in the AWS-managed `aws-s3` table bucket, making them queryable using SQL with AWS analytics services like Amazon Athena, Amazon Redshift, and Amazon EMR.

> **Note**
>
> There is no additional charge for exporting S3 Storage Lens metrics to AWS-managed S3 Tables. Standard charges apply for table storage, table management, and requests on the tables. For more information, see Amazon S3 pricing.

Enable S3 Tables export using the console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens**, and then choose **Storage Lens Dashboards**.
3. In **Storage Lens Dashboards** list, choose the dashboard that you want to edit.
4. Choose **Edit**.
5. On the **Dashboard** page, navigate to **Metrics export and publishing** section.
6. To enable Table Export for **Default metrics report**, select **Table bucket** in the Bucket type.
7. To enable Table Export for **Expanded prefixes metrics report**, select **Table bucket** in the Bucket type.
8. Review dashboard config and choose **Submit**.

> **Note**
>
> After you enable S3 Tables export, it can take up to 48 hours for the first data to be available in the tables.

> **Note**
>
> There is no additional charge for exporting S3 Storage Lens metrics to AWS-managed S3 Tables. Standard charges apply for table storage, table management, requests on the tables, and monitoring. You can enable or disable export to S3 Tables by using the Amazon S3 console, Amazon S3 API, the AWS CLI, or AWS SDKs.

> **Note**
>
> By default, records in your S3 tables don't expire. To help minimize storage costs for your tables, you can enable and configure record expiration for the tables. With this option, Amazon S3 automatically removes records from a table when the records expire. For more information, see Record expiration for tables.

Enable S3 Tables export using the AWS CLI

> **Note**
>
> Before running the following commands, make sure that you have an up to date CLI version. See Installing or updating to the latest version of the AWS CLI.

The following example enables S3 Tables export for an S3 Storage Lens configuration "Default metrics report" using the AWS CLI. To use this example, replace the *user input placeholders* with your own information.

```
aws s3control put-storage-lens-configuration --account-id=555555555555 --config-
id=your-configuration-id --storage-lens-configuration '{
   "Id":"your-configuration-id",
   "AccountLevel":{
      "ActivityMetrics":{
        "IsEnabled":true
      },
      "BucketLevel":{
         "ActivityMetrics":{
            "IsEnabled":true
         }
      }
   },
   "DataExport":{
      "S3BucketDestination":{
         "OutputSchemaVersion":"V_1",
         "Format":"CSV",
         "AccountId":"555555555555",
         "Arn":"arn:aws:s3:::my-export-bucket",
         "Prefix":"storage-lens-exports/"
      },
      "StorageLensTableDestination":{
         "IsEnabled":true
      }
   },
   "IsEnabled":true
}'
```

Enable S3 Tables export using the AWS SDKs

The following example enables S3 Tables export for an S3 Storage Lens configuration "Default metrics report" using the AWS SDK for Python (Boto3). To use this example, replace the *user input placeholders* with your own information.

```
import boto3

s3control = boto3.client('s3control')

response = s3control.put_storage_lens_configuration( AccountId='555555555555',
 ConfigId='your-configuration-id', StorageLensConfiguration={
        'Id': 'your-configuration-id',
        'AccountLevel': {
            'ActivityMetrics': {
              'IsEnabled': True
            },
            'BucketLevel': {
                'ActivityMetrics': {
                    'IsEnabled': True
                }
            }
        },
        'DataExport': {
            'S3BucketDestination': {
                'OutputSchemaVersion': 'V_1',
                'Format': 'CSV',
                'AccountId': '555555555555',
                'Arn': 'arn:aws:s3:::my-export-bucket',
                'Prefix': 'storage-lens-exports/'
            },
            'StorageLensTableDestination': {
                'IsEnabled': True
            }
        },
        'IsEnabled': True
    }
)
```

For more information about using the AWS SDKs, see AWS SDKs and tools.

Next steps

After enabling S3 Tables export, you can:

- Learn about Table naming for S3 Storage Lens export to S3 Tables
- Learn about Understanding S3 Storage Lens table schemas

Table naming for S3 Storage Lens export to S3 Tables

When you export S3 Storage Lens metrics to S3 Tables, the tables are organized using Apache Iceberg catalog conventions with specific naming patterns to ensure compatibility and organization.

Table location structure

The complete table location follows this pattern:

```
s3tablescatalog/aws-s3/<namespace>/<table-name>
```

Table bucket name

Table Bucket: `aws-s3`

The S3 Storage Lens export uses the `aws-s3` table bucket, which is the designated bucket for AWS S3-related system tables.

Catalog name

Catalog: `s3tablescatalog/aws-s3`

S3 Storage Lens tables are stored in the S3 catalog because Storage Lens provides insights about three types of S3 resources:

- Storage metrics
- Bucket properties
- API usage metrics

Namespace naming convention

Namespaces organize tables within the catalog. For S3 Storage Lens, the namespace is derived from your Storage Lens configuration ID.

Standard namespace format

For Storage Lens configuration IDs without dots (`.`):

```
lens_<configuration-id>_exp
```

Example: If your configuration ID is `my-lens-config`, the namespace will be:

```
lens_my-lens-config_exp
```

Namespace format with dot character or uppercase letters handling

Storage Lens configuration IDs can contain dots (`.`) or uppercase letters (`A-Z`), but S3 Tables namespaces only support lowercase letters, numbers, hyphens (`-`), and underscores (`_`). When your configuration ID contains dots, they are converted to hyphens, uppercase letters are converted to lower case letters, and a hash suffix is added for uniqueness:

```
lens_<configuration-id-with-dots-or-uppercase-replaced>_exp_<7-char-hash>
```

Example: If your configuration ID is `my.LENS.config`, the namespace will be:

```
lens_my-lens-config_exp_a1b2c3d
```

Where `a1b2c3d` is the first 7 characters of the SHA-1 hash of the original configuration ID.

Complete examples

For a Storage Lens configuration with ID `production-metrics`:

- **Table Bucket:** `aws-s3`
- **Catalog:** `s3tablescatalog/aws-s3`
- **Namespace:** `lens_production-metrics_exp`
- **Full Path:** `s3tablescatalog/aws-s3/lens_production-metrics_exp/<table-name>`

For a Storage Lens configuration with ID `prod.us.east.metrics`:

- **Table Bucket:** `aws-s3`

- **Catalog:** `s3tablescatalog/aws-s3`
- **Namespace:** `lens_prod-us-east-metrics_exp_f8e9a1b` (with hash)
- **Full Path:** `s3tablescatalog/aws-s3/lens_prod-us-east-metrics_exp_f8e9a1b/<table-name>`

Table types

The following table shows the different types of tables created for S3 Storage Lens exports:

Catalog	Namespace	S3 table name	Description
s3tablescatalog/aws-s3	lens_<conf_name>_exp[_<hash>]	default_storage_metrics	This table contains the storage metrics for your Storage Lens configuration.
s3tablescatalog/aws-s3	lens_<conf_name>_exp[_<hash>]	default_activity_metrics	This table contains the activity metrics for your Storage Lens configuration.
s3tablescatalog/aws-s3	lens_<conf_name>_exp[_<hash>]	expanded_prefixes_storage_metrics	This table contains the storage metrics for all the prefixes in your Storage Lens configuration.
s3tablescatalog/aws-s3	lens_<conf_name>_exp[_<hash>]	expanded_prefixes_activity_metrics	This table contains the activity metrics for all the prefixes in your Storage Lens configuration.
s3tablescatalog/aws-s3	lens_<conf_name>_exp[_<hash>]	bucket_property_metrics	This table contains the bucket property metrics for all the buckets in your Storage Lens configuration.

Next steps

- Learn about Understanding S3 Storage Lens table schemas
- Learn about Permissions for S3 Storage Lens tables

Understanding S3 Storage Lens table schemas

When exporting S3 Storage Lens metrics to S3 tables, the data is organized into three separate table schemas: storage metrics, bucket property metrics, and activity metrics.

Storage metrics table schema

Name	Type	Description
`version_number`	string	Version identifier of the schema of the table
`configuration_id`	string	S3 Storage Lens configuration name
`report_time`	timestamptz	Date the S3 Storage Lens report refers to
`aws_account_id`	string	Account id the entry refers to
`aws_region`	string	Region
`storage_class`	string	Storage Class
`record_type`	string	Type of record, related to what is the level of aggregation of data. Values: `ACCOUNT`, `BUCKET`, `PREFIX`, `STORAGE_LENS_GROUP_BUCKET`, `STORAGE_LENS_GROUP_ACCOUNT`.
`record_value`	string	Disambiguator for record types that have more than one record under them. It is used to reference the prefix
`bucket_name`	string	Bucket name
`object_count`	long	Number of objects stored for the current referenced item
`storage_bytes`	DECIMAL(38,0)	Number of bytes stored for the current referenced item

Name	Type	Description
bucket_key_sse_kms_object_count	long	Number of objects encrypted with a customer managed key stored for the current referenced item
bucket_key_sse_kms_storage_bytes	DECIMAL(38,0)	Number of bytes encrypted with a customer managed key stored for the current referenced item
current_version_object_count	long	Number of current version objects stored for the current referenced item
current_version_storage_bytes	DECIMAL(38,0)	Number of current version bytes stored for the current referenced item
delete_marker_object_count	long	Number of delete marker objects stored for the current referenced item
delete_marker_storage_bytes	DECIMAL(38,0)	Number of delete marker bytes stored for the current referenced item
encrypted_object_count	long	Number of encrypted objects stored for the current referenced item
encrypted_storage_bytes	DECIMAL(38,0)	Number of encrypted bytes stored for the current referenced item
incomplete_mpu_object_older_than_7_days_count	long	Number of incomplete multipart upload objects older than 7 days stored for the current referenced item

Name	Type	Description
incomplete_mpu_storage_older_than_7_days_bytes	DECIMAL(38,0)	Number of incomplete multipart upload bytes stored older than 7 days for the current referenced item
incomplete_mpu_object_count	long	Number of incomplete multipart upload objects stored for the current referenced item
incomplete_mpu_storage_bytes	DECIMAL(38,0)	Number of incomplete multipart upload bytes stored for the current referenced item
non_current_version_object_count	long	Number of non-current version objects stored for the current referenced item
non_current_version_storage_bytes	DECIMAL(38,0)	Number of non-current version bytes stored for the current referenced item
object_lock_enabled_object_count	long	Number of objects stored with lock enabled for the current referenced item
object_lock_enabled_storage_bytes	DECIMAL(38,0)	Number of bytes stored for objects with lock enabled in the current referenced item
replicated_object_count	long	Number of objects replicated for the current referenced item

Name	Type	Description
replicated_storage_bytes	DECIMAL(38,0)	Number of bytes replicated for the current referenced item
replicated_object_source_count	long	Number of objects replicated as source stored for the current referenced item
replicated_storage_source_bytes	DECIMAL(38,0)	Number of bytes replicated as source for the current referenced item
sse_kms_object_count	long	Number of objects encrypted with SSE key stored for the current referenced item
sse_kms_storage_bytes	DECIMAL(38,0)	Number of bytes encrypted with SSE key stored for the current referenced item
object_0kb_count	long	Number of objects with size equal to 0KB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_0kb_to_128kb_count	long	Number of objects with size greater than 0KB and less than or equal to 128KB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_128kb_to_256kb_count	long	Number of objects with size greater than 128KB and less than or equal to 256KB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_256kb_to_512kb_count	long	Number of objects with size greater than 256KB and less than or equal to 512KB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers

Name	Type	Description
object_512kb_to_1mb_count	long	Number of objects with size greater than 512KB and less than or equal to 1MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_1mb_to_2mb_count	long	Number of objects with size greater than 1MB and less than or equal to 2MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_2mb_to_4mb_count	long	Number of objects with size greater than 2MB and less than or equal to 4MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_4mb_to_8mb_count	long	Number of objects with size greater than 4MB and less than or equal to 8MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_8mb_to_16mb_count	long	Number of objects with size greater than 8MB and less than or equal to 16MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_16mb_to_32mb_count	long	Number of objects with size greater than 16MB and less than or equal to 32MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_32mb_to_64mb_count	long	Number of objects with size greater than 32MB and less than or equal to 64MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers

Name	Type	Description
object_64mb_to_128mb_count	long	Number of objects with size greater than 64MB and less than or equal to 128MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_128mb_to_256mb_count	long	Number of objects with size greater than 128MB and less than or equal to 256MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_256mb_to_512mb_count	long	Number of objects with size greater than 256MB and less than or equal to 512MB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_512mb_to_1gb_count	long	Number of objects with size greater than 512MB and less than or equal to 1GB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_1gb_to_2gb_count	long	Number of objects with size greater than 1GB and less than or equal to 2GB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_2gb_to_4gb_count	long	Number of objects with size greater than 2GB and less than or equal to 4GB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers
object_larger_than_4gb_count	long	Number of objects with size greater than 4GB, including current version, noncurrent versions, incomplete multipart uploads, and delete markers

Bucket property metrics table schema

Name	Type	Description
version_number	string	Version identifier of the schema of the table
configuration_id	string	S3 Storage Lens configuration name
report_time	timestamptz	Date the S3 Storage Lens report refers to
aws_account_id	string	Account id the entry refers to
record_type	string	Type of record, related to what is the level of aggregation of data. Values: ACCOUNT, BUCKET, PREFIX, STORAGE_LENS_GROUP_BUCKET , STORAGE_LENS_GROUP_ACCOUNT .
record_value	string	Disambiguator for record types that have more than one record under them. It is used to reference the prefix.
aws_region	string	Region
storage_class	string	Storage Class
bucket_name	string	Bucket name
versioning_enabled_bucket_count	long	Number of buckets with versioning enabled for the current referenced item
mfa_delete_enabled_bucket_count	long	Number of buckets with MFA delete enabled for the current referenced item
sse_kms_enabled_bucket_count	long	Number of buckets with KMS enabled for the current referenced item

Name	Type	Description
object_ownership_bucket_owner_enforced_bucket_count	long	Number of buckets with Object Ownership bucket owner enforced for the current referenced item
object_ownership_bucket_owner_preferred_bucket_count	long	Number of buckets with Object Ownership bucket owner preferred for the current referenced item
object_ownership_object_writer_bucket_count	long	Number of buckets with Object Ownership object writer for the current referenced item
transfer_acceleration_enabled_bucket_count	long	Number of buckets with transfer acceleration enabled for the current referenced item
event_notification_enabled_bucket_count	long	Number of buckets with event notification enabled for the current referenced item
transition_lifecycle_rule_count	long	Number of transition lifecycle rules for the current referenced item

Name	Type	Description
`expiration_lifecycle_rule_count`	long	Number of expiration lifecycle rules for the current referenced item
`non_current_version_transition_lifecycle_rule_count`	long	Number of noncurrent version transition lifecycle rules for the current referenced item
`non_current_version_expiration_lifecycle_rule_count`	long	Number of noncurrent version expiration lifecycle rules for the current referenced item
`abort_incomplete_multipart_upload_lifecycle_rule_count`	long	Number of abort incomplete multipart upload lifecycle rules for the current referenced item
`expired_object_delete_marker_lifecycle_rule_count`	long	Number of expire object delete marker lifecycle rules for the current referenced item
`same_region_replication_rule_count`	long	Number of Same-Region Replication rules for the current referenced item

Name	Type	Description
cross_region_replication_rule_count	long	Number of Cross-Region Replication rules for the current referenced item
same_account_replication_rule_count	long	Number of same-account replication rules for the current referenced item
cross_account_replication_rule_count	long	Number of cross-account replication rules for the current referenced item
invalid_destination_replication_rule_count	long	Number of buckets with invalid destination replication for the current referenced item

Activity metrics table schema

Name	Type	Description
version_number	string	Version identifier of the schema of the table
configuration_id	string	S3 Storage Lens configuration name
report_time	timestamptz	Date the S3 Storage Lens report refers to
aws_account_id	string	Account id the entry refers to

Name	Type	Description
aws_region	string	Region
storage_class	string	Storage Class
record_type	string	Type of record, related to what is the level of aggregation of data. Values: ACCOUNT, BUCKET, PREFIX, STORAGE_LENS_GROUP_BUCKET , STORAGE_LENS_GROUP_ACCOUNT .
record_value	string	Disambiguator for record types that have more than one record under them. It is used to reference the prefix
bucket_name	string	Bucket name
all_request_count	long	Number of all requests for the current referenced item
all_sse_kms_encrypted_request_count	long	Number of KMS encrypted requests for the current referenced item
all_unsupported_sig_request_count	long	Number of unsupported sig requests for the current referenced item
all_unsupported_tls_request_count	long	Number of unsupported TLS requests for the current referenced item
bad_request_error_400_count	long	Number of 400 bad request errors for the current referenced item

Name	Type	Description
delete_request_count	long	Number of delete requests for the current referenced item
downloaded_bytes	decimal(38,0)	Number of downloaded bytes for the current referenced item
error_4xx_count	long	Number of 4xx errors for the current referenced item
error_5xx_count	long	Number of 5xx errors for the current referenced item
forbidden_error_403_count	long	Number of 403 forbidden errors for the current referenced item
get_request_count	long	Number of get requests for the current referenced item
head_request_count	long	Number of head requests for the current referenced item
internal_server_error_500_count	long	Number of 500 internal server errors for the current referenced item
list_request_count	long	Number of list requests for the current referenced item
not_found_error_404_count	long	Number of 404 not found errors for the current referenced item
ok_status_200_count	long	Number of 200 OK requests for the current referenced item
partial_content_status_206_count	long	Number of 206 partial content requests for the current referenced item

Name	Type	Description
post_request_count	long	Number of post requests for the current referenced item
put_request_count	long	Number of put requests for the current referenced item
select_request_count	long	Number of select requests for the current referenced item
select_returned_bytes	decimal(38,0)	Number of bytes returned by select requests for the current referenced item
select_scanned_bytes	decimal(38,0)	Number of bytes scanned by select requests for the current referenced item
service_unavailable_error_503_count	long	Number of 503 service unavailable errors for the current referenced item
uploaded_bytes	decimal(38,0)	Number of uploaded bytes for the current referenced item
average_first_byte_latency	long	Average per-request time between when an S3 bucket receives a complete request and when it starts returning the response, measured over the past 24 hours
average_total_request_latency	long	Average elapsed per-request time between the first byte received and the last byte sent to an S3 bucket, measured over the past 24 hours
read_0kb_request_count	long	Number of GetObject requests with data sizes of 0KB, including both range-based requests and whole object requests

Name	Type	Description
read_0kb_to_128kb_request_count	long	Number of GetObject requests with data sizes greater than 0KB and up to 128KB, including both range-based requests and whole object requests
read_128kb_to_256kb_request_count	long	Number of GetObject requests with data sizes greater than 128KB and up to 256KB, including both range-based requests and whole object requests
read_256kb_to_512kb_request_count	long	Number of GetObject requests with data sizes greater than 256KB and up to 512KB, including both range-based requests and whole object requests
read_512kb_to_1mb_request_count	long	Number of GetObject requests with data sizes greater than 512KB and up to 1MB, including both range-based requests and whole object requests
read_1mb_to_2mb_request_count	long	Number of GetObject requests with data sizes greater than 1MB and up to 2MB, including both range-based requests and whole object requests
read_2mb_to_4mb_request_count	long	Number of GetObject requests with data sizes greater than 2MB and up to 4MB, including both range-based requests and whole object requests
read_4mb_to_8mb_request_count	long	Number of GetObject requests with data sizes greater than 4MB and up to 8MB, including both range-based requests and whole object requests
read_8mb_to_16mb_request_count	long	Number of GetObject requests with data sizes greater than 8MB and up to 16MB, including both range-based requests and whole object requests

Name	Type	Description
read_16mb_to_32mb_request_count	long	Number of GetObject requests with data sizes greater than 16MB and up to 32MB, including both range-based requests and whole object requests
read_32mb_to_64mb_request_count	long	Number of GetObject requests with data sizes greater than 32MB and up to 64MB, including both range-based requests and whole object requests
read_64mb_to_128mb_request_count	long	Number of GetObject requests with data sizes greater than 64MB and up to 128MB, including both range-based requests and whole object requests
read_128mb_to_256mb_request_count	long	Number of GetObject requests with data sizes greater than 128MB and up to 256MB, including both range-based requests and whole object requests
read_256mb_to_512mb_request_count	long	Number of GetObject requests with data sizes greater than 256MB and up to 512MB, including both range-based requests and whole object requests
read_512mb_to_1gb_request_count	long	Number of GetObject requests with data sizes greater than 512MB and up to 1GB, including both range-based requests and whole object requests
read_1gb_to_2gb_request_count	long	Number of GetObject requests with data sizes greater than 1GB and up to 2GB, including both range-based requests and whole object requests
read_2gb_to_4gb_request_count	long	Number of GetObject requests with data sizes greater than 2GB and up to 4GB, including both range-based requests and whole object requests

Name	Type	Description
read_larger_than_4gb_request_count	long	Number of GetObject requests with data sizes greater than 4GB, including both range-based requests and whole object requests
write_0kb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes of 0KB
write_0kb_to_128kb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 0KB and up to 128KB
write_128kb_to_256kb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 128KB and up to 256KB
write_256kb_to_512kb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 256KB and up to 512KB
write_512kb_to_1mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 512KB and up to 1MB
write_1mb_to_2mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 1MB and up to 2MB
write_2mb_to_4mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 2MB and up to 4MB

Name	Type	Description
write_4mb_to_8mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 4MB and up to 8MB
write_8mb_to_16mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 8MB and up to 16MB
write_16mb_to_32mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 16MB and up to 32MB
write_32mb_to_64mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 32MB and up to 64MB
write_64mb_to_128mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 64MB and up to 128MB
write_128mb_to_256mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 128MB and up to 256MB
write_256mb_to_512mb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 256MB and up to 512MB
write_512mb_to_1gb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 512MB and up to 1GB

Name	Type	Description
write_1gb_to_2gb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 1GB and up to 2GB
write_2gb_to_4gb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 2GB and up to 4GB
write_larger_than_4gb_request_count	long	Number of PutObject, UploadPart, and CreateMultipartUpload requests with data sizes greater than 4GB
concurrent_put_503_error_count	long	Number of 503 errors that are generated due to concurrent writes to the same object
cross_region_request_count	long	Number of requests that originate from a client in different Region than bucket's home Region
cross_region_transferred_bytes	decimal(38,0)	Number of bytes that are transferred from calls in different Region than bucket's home Region
cross_region_without_replication_request_count	long	Number of requests that originate from a client in different Region than bucket's home Region, excluding cross-region replication requests
cross_region_without_replication_transferred_bytes	decimal(38,0)	Number of bytes that are transferred from calls in different Region than bucket's home Region, excluding cross-region replication bytes

Name	Type	Description
`inregion_request_count`	long	Number of requests that originate from a client in same Region as bucket's home Region
`inregion_transferred_bytes`	decimal(38,0)	Number of bytes that are transferred from calls from same Region as bucket's home Region
`unique_objects_accessed_daily_count`	long	Number of objects that were accessed at least once in last 24 hrs

Next steps

- Learn about Permissions for S3 Storage Lens tables
- Start Querying S3 Storage Lens data with analytics tools
- Review the Amazon S3 Storage Lens metrics glossary for detailed metric definitions

Permissions for S3 Storage Lens tables

To work with S3 Storage Lens data exported to S3 Tables, you need appropriate AWS Identity and Access Management (IAM) permissions. This topic covers the permissions required for exporting metrics and managing encryption.

Permissions for metrics export to S3 Tables

To create and work with S3 Storage Lens tables and table buckets, you must have certain `s3tables` permissions. At a minimum, to configure S3 Storage Lens to S3 Tables, you must have the following `s3tables` permissions:

- `s3tables:CreateTableBucket` – This permission allows you to create an AWS-managed table bucket. All S3 Storage Lens metrics in your account are stored in a single AWS-managed table bucket named `aws-s3`.

- `s3tables:PutTableBucketPolicy` – S3 Storage Lens uses this permission to set a table bucket policy that allows `systemtables.s3.amazonaws.com` access to the bucket so that logs can be delivered.

> **Important**
>
> If you remove permissions for the service principal `systemtables.s3.amazonaws.com`, S3 Storage Lens will not be able to update the S3 tables with data based on your configuration. We recommend adding other access control policies in addition to the policy already provided, instead of editing the canned policy that is added when your table bucket is set up.

> **Note**
>
> A separate S3 table for each type of metric export is created for each Storage Lens configuration. If you have multiple Storage Lens configurations in the Region, separate tables are created for additional configurations. For example, there are three types of tables available for your S3 table bucket.

Permissions for AWS KMS encrypted tables

All data in S3 tables including S3 Storage Lens metrics are encrypted with SSE-S3 encryption by default. You can choose to encrypt your Storage Lens metrics report with AWS KMS keys (SSE-KMS). If you choose to encrypt your S3 Storage Lens metric reports with KMS keys, you must have additional permissions.

1. The user or IAM role needs the following permissions. You can grant these permissions by using the IAM console at https://console.aws.amazon.com/iam/.
   - `kms:DescribeKey` on the AWS KMS key used
2. On the key policy for the AWS KMS key, you need the following permissions. You can grant these permissions by using the AWS KMS console at https://console.aws.amazon.com/kms. To use this policy, replace the *`user input placeholders`* with your own information.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "EnableSystemTablesKeyUsage",
            "Effect": "Allow",
            "Principal": {
                "Service": "systemtables.s3.amazonaws.com"
            },
            "Action": [
                "kms:DescribeKey",
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-id",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "EnableKeyUsage",
            "Effect": "Allow",
            "Principal": {
                "Service": "maintenance.s3tables.amazonaws.com"
            },
            "Action": [
                "kms:GenerateDataKey",
                "kms:Decrypt"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/key-id",
            "Condition": {
                "StringLike": {
                    "kms:EncryptionContext:aws:s3:arn": "<table-bucket-arn>/*"
                }
            }
        }
    ]
}
```

Service-linked role for S3 Storage Lens

S3 Storage Lens uses a service-linked role to write metrics to S3 Tables. This role is automatically created when you enable S3 Tables export for the first time in your account. The service-linked role has the following permissions:

- `s3tables:CreateTable` - To create tables in the `aws-s3` table bucket
- `s3tables:PutTableData` - To write metrics data to tables
- `s3tables:GetTable` - To retrieve table metadata

You don't need to manually create or manage this service-linked role. For more information about service-linked roles, see Using service-linked roles in the *IAM User Guide*.

Best practices for permissions

Follow these best practices when configuring permissions for S3 Storage Lens tables:

- **Use least privilege** - Grant only the permissions required for specific tasks. For example, if users only need to query data, don't grant permissions to modify Storage Lens configurations.
- **Use IAM roles** - Use IAM roles instead of long-term access keys for applications and services that access S3 Storage Lens tables.
- **Enable AWS CloudTrail** - Enable CloudTrail logging to monitor access to S3 Storage Lens tables and track permission changes.
- **Use resource-based policies** - When possible, use resource-based policies to control access to specific tables or namespaces.
- **Regularly review permissions** - Periodically review and audit IAM policies and Lake Formation permissions to ensure they follow the principle of least privilege.

Troubleshooting permissions

Access denied when enabling S3 Tables export

Problem: You receive an "access denied" error when trying to enable S3 Tables export.

Solution: Verify that your IAM user or role has the `s3:PutStorageLensConfiguration` permission and the necessary S3 Tables permissions.

Access denied when querying tables

Problem: You receive an "access denied" error when querying S3 Storage Lens tables in Amazon Athena.

Solution: Verify that:

- Analytics integration is enabled on the `aws-s3` table bucket
- Lake Formation permissions are correctly configured
- Your IAM user or role has the necessary Amazon Athena permissions

KMS encryption errors

Problem: You receive KMS-related errors when accessing encrypted tables.

Solution: Verify that:

- Your IAM policy includes the required KMS permissions
- The KMS key policy grants permissions to the S3 Storage Lens service principal
- The KMS key is in the same Region as your Storage Lens configuration

Next steps

- Learn about Setting Amazon S3 Storage Lens permissions
- Learn about Querying S3 Storage Lens data with analytics tools
- Learn about Using AI assistants with S3 Storage Lens tables

Querying S3 Storage Lens data with analytics tools

Before you can query S3 Storage Lens data exported to S3 Tables using AWS analytics services like Amazon Athena or Amazon EMR, you must enable analytics integration on the AWS-managed `aws-s3` table bucket and configure AWS Lake Formation permissions.

Important

Enabling analytics integration on the "aws-s3" table bucket is a required step that is often missed. Without this configuration, you will not be able to query your S3 Storage Lens tables using AWS analytics services.

Prerequisites

Before you begin, ensure that you have:

- An S3 Storage Lens configuration with S3 Tables export enabled. For more information, see Exporting S3 Storage Lens metrics to S3 Tables .
- Access to Amazon Athena or another analytics service.
- Waited 24-48 hours after enabling export for the first data to be available.

Integration overview

For detailed information about integrating S3 Tables with AWS analytics services, including prerequisites, IAM role configuration, and step-by-step procedures, see Integrating Amazon S3 Tables with AWS analytics services.

After you enable S3 Tables export and set up analytics integration, you can query your S3 Storage Lens data using AWS analytics services such as Amazon Athena, Amazon Redshift, and Amazon EMR. This enables you to perform custom analysis, create dashboards, and derive insights from your storage data using standard SQL.

Querying with Amazon Athena

Amazon Athena is a serverless interactive query service that makes it easy to analyze data using standard SQL. Use the following steps to query S3 Storage Lens data in Athena.

Note

In all query examples, replace `lens_my-config_exp` with your actual Storage Lens configuration namespace. For more information about namespace naming, see Table naming for S3 Storage Lens export to S3 Tables .

Example: Query top storage consumers

The following query identifies the top 10 buckets by storage consumption:

```
SELECT
    bucket_name,
    storage_class,
    SUM(storage_bytes) / POWER(1024, 3) AS storage_gb,
    SUM(object_count) AS objects
FROM "s3tablescatalog/aws-s3"."lens_my-config_exp"."default_storage_metrics"
WHERE report_time = (
    SELECT MAX(report_time)
    FROM "s3tablescatalog/aws-s3"."lens_my-config_exp"."default_storage_metrics"
)
    AND record_type = 'BUCKET'
    AND bucket_name != ''
GROUP BY bucket_name, storage_class
ORDER BY storage_gb DESC
LIMIT 10
```

Example: Analyze storage growth over time

The following query analyzes storage growth over the last 30 days:

```
SELECT
    CAST(report_time AS date) AS report_date,
    SUM(storage_bytes) / POWER(1024, 3) AS total_storage_gb
FROM "s3tablescatalog/aws-s3"."lens_my-config_exp"."default_storage_metrics"
WHERE report_time >= current_date - interval '30' day
    AND record_type = 'ACCOUNT'
GROUP BY CAST(report_time AS date)
ORDER BY report_date DESC;
```

Example: Identify incomplete multipart uploads

The following query finds buckets with incomplete multipart uploads older than 7 days:

```
SELECT
    bucket_name,
    SUM(incomplete_mpu_storage_older_than_7_days_bytes) / POWER(1024, 3) AS
 wasted_storage_gb,
    SUM(incomplete_mpu_object_older_than_7_days_count) AS wasted_objects
FROM "s3tablescatalog/aws-s3"."lens_my-config_exp"."default_storage_metrics"
```

```
WHERE report_time = (
    SELECT MAX(report_time)
    FROM "s3tablescatalog/aws-s3"."lens_my-config_exp"."default_storage_metrics"
)
    AND record_type = 'BUCKET'
    AND incomplete_mpu_storage_older_than_7_days_bytes > 0
GROUP BY bucket_name
ORDER BY wasted_storage_gb DESC;
```

Example: Find cold data candidates

The following query identifies prefixes with no activity in the last 100 days that are stored in hot storage tiers:

```
WITH recent_activity AS (
    SELECT DISTINCT
        bucket_name,
        record_value AS prefix_path
    FROM "s3tablescatalog/aws-s3"."lens_my-
config_exp"."expanded_prefixes_activity_metrics"
    WHERE report_time >= current_date - interval '100' day
        AND record_type = 'PREFIX'
        AND all_request_count > 0
)
SELECT
    s.bucket_name,
    s.record_value AS prefix_path,
    s.storage_class,
    SUM(s.storage_bytes) / POWER(1024, 3) AS storage_gb
FROM "s3tablescatalog/aws-s3"."lens_my-config_exp"."expanded_prefixes_storage_metrics"
 s
LEFT JOIN recent_activity r
    ON s.bucket_name = r.bucket_name
    AND s.record_value = r.prefix_path
WHERE s.report_time = (
    SELECT MAX(report_time)
    FROM "s3tablescatalog/aws-s3"."lens_my-
config_exp"."expanded_prefixes_storage_metrics"
)
    AND s.record_type = 'PREFIX'
    AND s.storage_class IN ('STANDARD', 'REDUCED_REDUNDANCY')
    AND s.storage_bytes > 1073741824  -- > 1GB
    AND r.prefix_path IS NULL  -- No recent activity
```

```
GROUP BY s.bucket_name, s.record_value, s.storage_class
ORDER BY storage_gb DESC
LIMIT 20;
```

Example: Analyze request patterns

The following query analyzes request patterns to understand access frequency:

```
SELECT
    bucket_name,
    SUM(all_request_count) AS total_requests,
    SUM(get_request_count) AS get_requests,
    SUM(put_request_count) AS put_requests,
    ROUND(100.0 * SUM(get_request_count) / NULLIF(SUM(all_request_count), 0), 2) AS
 get_percentage,
    SUM(downloaded_bytes) / POWER(1024, 3) AS downloaded_gb
FROM "s3tablescatalog/aws-s3"."lens_my-config_exp"."default_activity_metrics"
WHERE report_time >= current_date - interval '7' day
    AND record_type = 'BUCKET'
    AND bucket_name != ''
GROUP BY bucket_name
HAVING SUM(all_request_count) > 0
ORDER BY total_requests DESC
LIMIT 10;
```

Querying with Apache Spark on Amazon EMR

Amazon EMR provides a managed Hadoop framework that makes it easy to process vast amounts of data using Apache Spark. You can use the Iceberg connector to read S3 Storage Lens tables directly.

Read S3 Tables with Spark

Use the following Python code to read S3 Storage Lens data with Spark:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("S3StorageLensAnalysis") \
    .config("spark.sql.catalog.s3tablescatalog",
 "org.apache.iceberg.spark.SparkCatalog") \
    .config("spark.sql.catalog.s3tablescatalog.catalog-impl",
 "org.apache.iceberg.aws.glue.GlueCatalog") \
```

```
    .getOrCreate()

# Read S3 Storage Lens data
df = spark.read \
    .format("iceberg") \
    .load("s3tablescatalog/aws-s3.lens_my-config_exp.default_storage_metrics")

# Analyze data
df.filter("record_type = 'BUCKET'") \
    .groupBy("bucket_name", "storage_class") \
    .sum("storage_bytes") \
    .orderBy("sum(storage_bytes)", ascending=False) \
    .show(10)
```

Query optimization best practices

Follow these best practices to optimize query performance and reduce costs:

- **Filter by report_time** – Always include date filters to reduce the amount of data scanned. This is especially important for tables with long retention periods.

  ```
  WHERE report_time >= current_date - interval '7' day
  ```

- **Use record_type filters** – Specify the appropriate aggregation level (ACCOUNT, BUCKET, PREFIX) to query only the data you need.

  ```
  WHERE record_type = 'BUCKET'
  ```

- **Include LIMIT clauses** – Use LIMIT for exploratory queries to control result size and reduce query costs.

  ```
  LIMIT 100
  ```

- **Filter empty records** – Use conditions to exclude empty or zero-value records.

  ```
  WHERE storage_bytes > 0
  ```

- **Use the latest data** – When analyzing current state, filter for the most recent report_time to avoid scanning historical data.

  ```
  WHERE report_time = (SELECT MAX(report_time) FROM table_name)
  ```

Example optimized query pattern

The following query demonstrates best practices for optimization:

```
SELECT
    bucket_name,
    SUM(storage_bytes) / POWER(1024, 3) AS storage_gb
FROM "s3tablescatalog/aws-s3"."lens_my-config_exp"."default_storage_metrics"
WHERE report_time >= current_date - interval '7' day  -- Date filter
    AND record_type = 'BUCKET'                         -- Record type filter
    AND storage_bytes > 0                              -- Non-empty filter
    AND bucket_name != ''                              -- Non-empty filter
GROUP BY bucket_name
ORDER BY storage_gb DESC
LIMIT 100;                                             -- Result limit
```

Troubleshooting

Query returns no results

Problem: Your query completes successfully but returns no results.

Solution:

- Verify that data is available by checking the latest report_time:

  ```
  SELECT MAX(report_time) AS latest_data
  FROM "s3tablescatalog/aws-s3"."lens_my-config_exp"."default_storage_metrics";
  ```

- Ensure that you're using the correct namespace name. Use `SHOW TABLES IN \`lens_my-config_exp\`;` to list available tables.
- Wait 24-48 hours after enabling S3 Tables export for the first data to be available.

Access denied errors

Problem: You receive access denied errors when running queries.

Solution: Verify that AWS Lake Formation permissions are correctly configured. For more information, see Integrating Amazon S3 Tables with AWS analytics services.

Next steps

- Learn about Using AI assistants with S3 Storage Lens tables
- Review the Amazon S3 Storage Lens metrics glossary for metric definitions
- Explore Amazon S3 Storage Lens metrics use cases for more analysis ideas
- Learn about Amazon Athena for serverless querying

Using AI assistants with S3 Storage Lens tables

You can use AI assistants and conversational AI tools to interact with your S3 Storage Lens data exported to S3 Tables using natural language. By leveraging the Model Context Protocol (MCP) and the MCP Server for Amazon S3 Tables, you can query, analyze, and gain insights from your storage data without writing SQL queries.

Overview

Model Context Protocol (MCP) is a standardized way for AI applications to access and utilize contextual information. The MCP Server for Amazon S3 Tables provides tools that enable AI assistants to interact with your S3 Tables data using natural language interfaces. This democratizes data access and enables individuals across technical skill levels to work with S3 Storage Lens metrics.

With the MCP Server for S3 Tables, you can use natural language to:

- List S3 table buckets, namespaces, and tables
- Query S3 Storage Lens metrics and get insights
- Analyze storage trends and patterns
- Identify cost optimization opportunities
- Generate reports and visualizations

Supported AI assistants

The MCP Server for S3 Tables works with various AI assistants that support the Model Context Protocol, including:

- **Kiro** - An AI coding assistant with built-in MCP support
- **Amazon Q Developer** - AWS's AI-powered assistant for developers

- **Cline** - An AI coding assistant with MCP integration
- **Claude Desktop** - Anthropic's desktop application with MCP support
- **Cursor** - An AI-powered code editor

> **Important**
>
> AI-generated SQL queries and recommendations should be reviewed and validated before use. Verify that queries are appropriate for your data structure, use case, and performance requirements. Always test recommendations in a non-production environment before implementing them in production.

Setting up Kiro with S3 Storage Lens tables

Kiro is an AI coding assistant that provides seamless integration with S3 Tables through the MCP Server. Kiro can help you install and configure the MCP Server directly through its interface, simplifying the setup process.

For more information about Kiro, see Kiro AI.

Prerequisites

Before you begin, ensure that you have:

- Kiro installed on your system. Download from https://kiro.ai/
- AWS CLI configured with appropriate credentials
- An S3 Storage Lens configuration with S3 Tables export enabled
- Permissions to query S3 Tables. For more information, see Permissions for S3 Storage Lens tables.

Step 1: Install the S3 Tables MCP Server

You can install the S3 Tables MCP Server in two ways:

Option 1: Using Kiro's built-in MCP server management

Kiro can help you discover and install MCP servers directly through its interface:

1. Open Kiro
2. Access the MCP server management interface (typically through settings or command palette)
3. Search for "S3 Tables" or "awslabs.s3-tables-mcp-server"
4. Follow Kiro's prompts to install and configure the server

Option 2: Manual installation using uvx

Alternatively, you can manually install the MCP Server using `uvx`, a Python package runner:

```
uvx awslabs.s3-tables-mcp-server@latest
```

For more information about installing the MCP Server, see the AWS S3 Tables MCP Server documentation.

Step 2: Configure Kiro MCP settings

Create or update your Kiro MCP configuration file at `~/.kiro/settings/mcp.json` with the following content:

```
{
  "mcpServers": {
    "awslabs.s3-tables-mcp-server": {
      "command": "uvx",
      "args": ["awslabs.s3-tables-mcp-server@latest"],
      "env": {
        "AWS_PROFILE": "your-aws-profile",
        "AWS_REGION": "us-east-1"
      }
    }
  }
}
```

Replace `your-aws-profile` with your AWS CLI profile name and `us-east-1` with your AWS Region.

Step 3: Verify the configuration

After configuring the MCP Server, restart Kiro and verify that the S3 Tables tools are available. You can check the available MCP servers in Kiro's settings or by asking Kiro to list available tools.

Example use cases with AI assistants

The following examples demonstrate how to use natural language prompts with AI assistants to interact with S3 Storage Lens data.

Example 1: Query top storage consumers

Prompt: "Show me the top 10 buckets by storage consumption from my S3 Storage Lens data."

The AI assistant will use the MCP Server to query your S3 Storage Lens tables and return the results, including bucket names, storage classes, and storage amounts.

Example 2: Analyze storage growth

Prompt: "Analyze my storage growth over the last 30 days and show me the trend."

The AI assistant will query the storage metrics table, calculate daily storage totals, and present the growth trend.

Example 3: Identify cost optimization opportunities

Prompt: "Find buckets with incomplete multipart uploads older than 7 days that are wasting storage."

The AI assistant will query the storage metrics table for incomplete multipart uploads and provide a list of buckets with potential cost savings.

Example 4: Find cold data candidates

Prompt: "Identify prefixes with no activity in the last 100 days that are stored in hot storage tiers."

The AI assistant will analyze both storage and activity metrics to identify data that could be moved to colder storage tiers for cost optimization.

Example 5: Generate storage reports

Prompt: "Create a summary report of my S3 storage showing total storage, object counts, and request patterns for the last week."

The AI assistant will query multiple tables, aggregate the data, and generate a comprehensive report.

Best practices for using AI assistants

Follow these best practices when using AI assistants with S3 Storage Lens data:

- **Be specific in your prompts** - Provide clear, specific instructions about what data you want to analyze and what insights you're looking for.
- **Verify AI-generated queries** - Always review and validate the SQL queries and recommendations that the AI assistant generates before executing them or taking action. AI assistants may occasionally produce incorrect queries or recommendations that need to be verified against your specific use case and data.
- **Use appropriate permissions** - Ensure that the IAM credentials used by the AI assistant have only the necessary permissions. For read-only analysis, grant only SELECT permissions.
- **Monitor usage** - Track the queries executed by AI assistants using AWS CloudTrail to maintain audit trails.
- **Start with simple queries** - Begin with straightforward queries to understand how the AI assistant interprets your prompts, then progress to more complex analysis.

Logging and traceability

When using the S3 Tables MCP Server with AI assistants, you have multiple ways to audit operations:

- **Local logs** - The MCP Server logs requests and responses locally. You can specify a log directory using the `--log-dir` configuration option.
- **AWS CloudTrail** - All S3 Tables operations via the MCP Server using PyIceberg will have `awslabs/mcp/s3-tables-mcp-server/<version>` as the user agent string. You can filter CloudTrail logs by this user agent to trace actions performed by AI assistants.
- **AI assistant history** - AI assistants like Kiro and Cline maintain history logs that record natural language requests, LLM responses, and instructions provided to the MCP Server.

Security considerations

When using AI assistants with S3 Storage Lens data, follow these security best practices:

- **Use least privilege access** - Grant AI assistants only the minimum permissions required for their tasks.
- **Enable MFA** - Use multi-factor authentication for AWS accounts that AI assistants access.

- **Review permissions regularly** - Periodically audit the permissions granted to AI assistants and revoke unnecessary access.
- **Use separate credentials** - Consider using separate AWS credentials for AI assistant access to facilitate tracking and auditing.
- **Avoid sharing sensitive data** - Be cautious about sharing sensitive information in prompts to AI assistants, especially when using cloud-based AI services.

Troubleshooting

AI assistant cannot connect to S3 Tables

Problem: The AI assistant reports that it cannot connect to S3 Tables or the MCP Server is not responding.

Solution:

- Verify that the MCP Server is correctly installed using `uvx awslabs.s3-tables-mcp-server@latest --version`
- Check that your AWS credentials are configured correctly
- Ensure that the MCP configuration file has the correct AWS profile and region

Access denied errors

Problem: The AI assistant receives access denied errors when querying S3 Storage Lens tables.

Solution:

- Verify that analytics integration is enabled on the `aws-s3` table bucket
- Check that Lake Formation permissions are correctly configured
- Ensure that the AWS credentials have the necessary IAM permissions

Incorrect or unexpected results

Problem: The AI assistant returns incorrect or unexpected results.

Solution:

- Review the SQL query generated by the AI assistant

- Verify that you're using the correct namespace name for your Storage Lens configuration
- Check that data is available by querying the latest report_time
- Refine your prompt to be more specific about what you want to analyze

Additional resources

For more information about using AI assistants with S3 Tables, see the following resources:

- Kiro AI - AI coding assistant with built-in MCP support
- Implementing conversational AI for S3 Tables using Model Context Protocol (MCP) - AWS Storage Blog
- AWS S3 Tables MCP Server documentation
- Model Context Protocol specification

Using Amazon S3 Storage Lens with AWS Organizations

Amazon S3 Storage Lens is a cloud-storage analytics feature that you can use to gain organization-wide visibility into object-storage usage and activity. You can use S3 Storage Lens metrics to generate summary insights, such as finding out how much storage you have across your entire organization or which are the fastest-growing buckets and prefixes. You can also use Amazon S3 Storage Lens to collect storage metrics and usage data for all AWS accounts that are part of your AWS Organizations hierarchy. To do this, you must be using AWS Organizations, and you must enable S3 Storage Lens trusted access by using your AWS Organizations management account.

After enabling trusted access, add delegated administrator access to accounts in your organization. The delegated administrator accounts are used to create S3 Storage Lens configurations and dashboards that collect organization-wide storage metrics and user data. For more information about enabling trusted access, see Amazon S3 Storage Lens and AWS Organizations in the *AWS Organizations User Guide*.

Topics

- Enabling trusted access for S3 Storage Lens
- Disabling trusted access for S3 Storage Lens
- Registering a delegated administrator for S3 Storage Lens
- Deregistering a delegated administrator for S3 Storage Lens

Enabling trusted access for S3 Storage Lens

By enabling trusted access, you allow Amazon S3 Storage Lens to access your AWS Organizations hierarchy, membership, and structure through AWS Organizations API operations. S3 Storage Lens then becomes a trusted service for your entire organization's structure.

Whenever a dashboard configuration is created, S3 Storage Lens creates service-linked roles in your organization's management or delegated administrator accounts. The service-linked role grants S3 Storage Lens permission to perform the following actions:

- Describe organizations
- List accounts
- Verify a list of AWS service access for the organizations
- Get delegated administrators for the organizations

S3 Storage Lens can then ensure that it has access to collect the cross-account metrics for the accounts in your organization. For more information, see Using service-linked roles for Amazon S3 Storage Lens.

After enabling trusted access, you can assign delegated administrator access to accounts in your organization. When an account is marked as a delegated administrator for a service, the account receives authorization to access all read-only organization API operations. This access provides the delegated administrator visibility to the members and structures of your organization so that they too can create S3 Storage Lens dashboards.

Note

- Trusted access can only be enabled by the management account.
- Only the management account and delegated administrators can create S3 Storage Lens dashboards or configurations for your organization.

Using the S3 console

To enable S3 Storage Lens to have AWS Organizations trusted access

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. On the left navigation pane, navigate to **Storage Lens**.
3. Choose **AWS Organizations settings**. The **AWS Organizations access for Storage Lens** page displays.
4. Under **AWS Organizations trusted access**, choose **Edit**.

   The **AWS Organizations access** page displays.
5. Choose **Enable** to enable trusted access for your S3 Storage Lens dashboard.
6. Choose **Save changes**.

Using the AWS CLI

Example

The following example shows you how to enable AWS Organizations trusted access for S3 Storage Lens in AWS CLI.

```
aws organizations enable-aws-service-access --service-principal storage-
lens.s3.amazonaws.com
```

Using the AWS SDK for Java

Example– Enable AWS Organizations trusted access for S3 Storage Lens using SDK for Java

The following example shows you how to enable trusted access for S3 Storage Lens in SDK for Java. To use this example, replace the *`user input placeholders`* with your own information.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
```

```
import com.amazonaws.services.organizations.model.EnableAWSServiceAccessRequest;

public class EnableOrganizationsTrustedAccess {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.enableAWSServiceAccess(new
 EnableAWSServiceAccessRequest()
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

Disabling trusted access for S3 Storage Lens

Removing an account as a delegated administrator or disabling trusted access limits the account owner's S3 Storage Lens dashboard metrics to work only on an account level. Each account holder is then only be able to see the benefits of S3 Storage Lens through the limited scope of their account, and not their entire organization.

When you disable trusted access in S3 Storage Lens, any dashboards requiring trusted access are no longer updated. Any organizational dashboards that are created are also no longer updated. Instead, you're only able to query historic data for the S3 Storage Lens dashboard, while the data is still available.

Note

- Disabling trusted access for S3 Storage Lens also automatically stops all organization-level dashboards from collecting and aggregating storage metrics. This is because S3 Storage Lens no longer has trusted access to the organization accounts.
- Your management and delegate administrator accounts can still see the historic data for any disabled dashboards. They can also query this historic data while it is still available.

Using the S3 console

To disable trusted access for S3 Storage Lens

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. On the left navigation pane, navigate to **Storage Lens**.
3. Choose **AWS Organizations settings**. The **AWS Organizations access for Storage Lens** page displays.
4. Under **AWS Organizations trusted access**, choose **Edit**.

   The **AWS Organizations access** page displays.
5. Choose **Disable** to disable trusted access for your S3 Storage Lens dashboard.
6. Choose **Save changes**.

Using the AWS CLI

Example

The following example disables trusted access for S3 Storage Lens using the AWS CLI.

```
aws organizations disable-aws-service-access --service-principal storage-
lens.s3.amazonaws.com
```

Using the AWS SDK for Java

Example– Disable AWS Organizations trusted access for S3 Storage Lens

The following example shows you how to disable AWS Organizations trusted access for S3 Storage Lens in SDK for Java. To use this example, replace the *user input placeholders* with your own information.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import com.amazonaws.services.organizations.model.DisableAWSServiceAccessRequest;

public class DisableOrganizationsTrustedAccess {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            // Make sure to remove any existing delegated administrator for S3 Storage
 Lens
            // before disabling access; otherwise, the request will fail.
            organizationsClient.disableAWSServiceAccess(new
 DisableAWSServiceAccessRequest()
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
```

```
  }
}
```

Registering a delegated administrator for S3 Storage Lens

You can create organization-level dashboards by using your organization's management account or delegated administrator accounts. Delegated administrator accounts allow other accounts besides your management account to create organization-level dashboards. Only the management account of an organization can register and deregister other accounts as delegated administrators for the organization.

After enabling trusted access, you can register delegate administrator access to accounts in your organization by using the AWS Organizations REST API, AWS CLI, or SDKs from the management account. (For more information, see RegisterDelegatedAdministrator in the *AWS Organizations API Reference*.) When an account is registered as a delegated administrator, the account receives authorization to access all read-only AWS Organizations API operations. This provides visibility to the members and structures of your organization so that they can create S3 Storage Lens dashboards on your behalf.

> **Note**
>
> Before you can designate a delegated administrator by using the AWS Organizations REST API, AWS CLI, or SDKs, you must call the EnableAWSServiceAccess operation.

Using the S3 console

To register delegated administrators for S3 Storage Lens

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. On the left navigation pane, navigate to **Storage Lens**.
3. Choose **AWS Organizations settings**.
4. Under **Delegated administrators**, choose **Register account**.
5. Add an AWS account ID to register the account as a delegated administrator. The delegated administrator is able to create organization-level dashboards for all accounts and storage in your organization.
6. Choose **Register account**.

Using the AWS CLI

Example

The following example shows you how to register Organizations delegated administrators for S3 Storage Lens using the AWS CLI. To use this example, replace the *user input placeholders* with your own information.

```
aws organizations register-delegated-administrator --service-principal storage-
lens.s3.amazonaws.com --account-id 111122223333
```

Using the AWS SDK for Java

Example– Register Organizations delegated administrators for S3 Storage Lens

The following example shows you how to register AWS Organizations delegated administrators for S3 Storage Lens in SDK for Java. To use this example, replace the *user input placeholders* with your own information.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.RegisterDelegatedAdministratorRequest;

public class RegisterOrganizationsDelegatedAdministrator {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
 delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.registerDelegatedAdministrator(new
 RegisterDelegatedAdministratorRequest()
```

```
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

Deregistering a delegated administrator for S3 Storage Lens

After enabling trusted access, you can also deregister delegate administrator access to accounts in your organization. Delegated administrator accounts allow other accounts besides your management account to create organization-level dashboards. Only the management account of an organization can deregister accounts as delegated administrators for the organization.

You can deregister a delegated administrator by using the AWS Organizations AWS Management Console, REST API, AWS CLI, or AWS SDKS from the management account. For more information, see DeregisterDelegatedAdministrator in the *AWS Organizations API Reference*.

When an account is deregistered as a delegated administrator, the account loses access to the following:

- All read-only AWS Organizations API operations that provide visibility to the members and structures of your organization.
- All organization-level dashboards created by the delegated administrator. Deregistering a delegated administrator also automatically stops all organization-level dashboards created by that delegated administrator from aggregating new storage metrics.

> **Note**
>
> The deregistered delegated administrator will still be able to see the historic data for the disabled dashboards that they created if data is still available for querying.

Using the S3 console

To deregister delegated administrators for S3 Storage Lens

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. On the left navigation pane, navigate to **Storage Lens**.
3. Choose **AWS Organizations settings**.
4. Under **Delegated administrators**, choose the account that you wish to deregister.
5. Choose **De-register account**. The deregistered account is no longer a delegated administrator and is now unable to create organization-level dashboards for all accounts and storage in your organization.
6. Choose **Register account**.

Using the AWS CLI

Example

The following example shows you how to deregister Organizations delegated administrators for S3 Storage Lens using the AWS CLI. To use this example, replace *111122223333* with your own AWS account ID.

```
aws organizations deregister-delegated-administrator --service-principal storage-
lens.s3.amazonaws.com --account-id 111122223333
```

Using the AWS SDK for Java

Example– Deregister Organizations delegated administrators for S3 Storage Lens

The following example shows you how to deregister Organizations delegated administrators for S3 Storage Lens using SDK for Java. To use this example, replace the *user input placeholders* with your own information.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.organizations.AWSOrganizations;
```

```
import com.amazonaws.services.organizations.AWSOrganizationsClient;
import
 com.amazonaws.services.organizations.model.DeregisterDelegatedAdministratorRequest;

public class DeregisterOrganizationsDelegatedAdministrator {
 private static final String S3_STORAGE_LENS_SERVICE_PRINCIPAL = "storage-
lens.s3.amazonaws.com";

 public static void main(String[] args) {
  try {
            String delegatedAdminAccountId = "111122223333"; // Account Id for the
 delegated administrator.
            AWSOrganizations organizationsClient = AWSOrganizationsClient.builder()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(Regions.US_EAST_1)
                .build();

            organizationsClient.deregisterDelegatedAdministrator(new
 DeregisterDelegatedAdministratorRequest()
                .withAccountId(delegatedAdminAccountId)
                .withServicePrincipal(S3_STORAGE_LENS_SERVICE_PRINCIPAL));
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but AWS Organizations couldn't
 process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // AWS Organizations couldn't be contacted for a response, or the client
            // couldn't parse the response from AWS Organizations.
            e.printStackTrace();
        }
 }
}
```

Working with S3 Storage Lens groups to filter and aggregate metrics

An Amazon S3 Storage Lens group aggregates metrics using custom filters based on object metadata. Storage Lens groups help you drill down into characteristics of your data, such as distribution of objects by age, your most common file types, and more. For example, you can filter metrics by object tag to identify your fastest-growing datasets, or visualize your storage based on object size and age to inform your storage archive strategy. As a result, Amazon S3 Storage Lens groups helps you to better understand and optimize your S3 storage.

When you use Storage Lens groups, you can analyze and filter S3 Storage Lens metrics using object metadata such as prefixes, suffixes, object tags, object size, or object age. You can also apply a combination of these filters. After you attach your Storage Lens group to your S3 Storage Lens dashboard, you can view S3 Storage Lens metrics aggregated by Amazon S3 Storage Lens groups directly in your dashboard.

For example, you can also filter your metrics by object size or age bands to determine which portion of your storage consists of small objects. You can then use this information with S3 Intelligent-Tiering or S3 Lifecycle to transition small objects to different storage classes for cost and storage optimization.

Topics

- How S3 Storage Lens groups work
- Using Storage Lens groups

How S3 Storage Lens groups work

You can use Storage Lens groups to aggregate metrics using custom filters based on object metadata. When you define a custom filter, you can use prefixes, suffixes, object tags, object sizes, object age, or a combination of these custom filters. During Storage Lens group creation, you can also include a single filter or multiple filter conditions. To specify multiple filter conditions, you use `And` or `Or` logical operators.

When you create and configure a Storage Lens group, the Storage Lens group itself acts as a custom filter in the dashboard that you attach the group to. In your dashboard, you can then use the Storage Lens group filter to obtain storage metrics based on the custom filter that you defined in the group.

To view the data for your Storage Lens group in your S3 Storage Lens dashboard, you must attach the group to the dashboard after you've created the group. After your Storage Lens group is attached to your Storage Lens dashboard, your dashboard will collect storage usage metrics within 48 hours. You can then visualize this data in the Storage Lens dashboard or export it through a metrics export. If you forget to attach a Storage Lens group to a dashboard, your Storage Lens group data won’t be captured or displayed anywhere.

Note

- When you create a S3 Storage Lens group, you're creating an AWS resource. Therefore, each Storage Lens group has its own Amazon Resource Name (ARN), which you can specify when attaching it to or excluding it from a S3 Storage Lens dashboard.
- If your Storage Lens group isn't attached to a dashboard, you won't incur any additional charges for creating a Storage Lens group.
- S3 Storage Lens aggregates usage metrics for an object under all matching Storage Lens groups. Therefore, if an object matches the filter conditions for two or more Storage Lens groups, you will see repeated counts for the same object across your storage usage.

You can create a Storage Lens group at the account level in a specified home Region (from the list of supported AWS Regions). Then, you can attach your Storage Lens group to multiple Storage Lens dashboards, as long as the dashboards are in the same AWS account and home Region. You can create up to 50 Storage Lens groups per home Region in each AWS account.

You can create and manage S3 Storage Lens groups by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), AWS SDKs, or the Amazon S3 REST API.

Topics

- Viewing Storage Lens group aggregated metrics
- Storage Lens groups permissions
- Storage Lens groups configuration
- AWS resource tags
- Storage Lens groups metrics export

Viewing Storage Lens group aggregated metrics

You can view the aggregated metrics for your Storage Lens groups by attaching the groups to a dashboard. The Storage Lens groups that you want to attach must reside within the designated home Region in the dashboard account.

To attach a Storage Lens group to a dashboard, you must specify the group in the **Storage Lens group aggregation** section of your dashboard configuration. If you have several Storage Lens

groups, you can filter the **Storage Lens group aggregation** results to include or exclude only the groups that you want. For more information about attaching groups to your dashboards, see the section called "Attach or remove a Storage Lens group".

After you've attached your groups, you will see the additional Storage Lens group aggregation data in your dashboard within 48 hours.

> **Note**
>
> To view aggregated metrics for your Storage Lens group, you must attach the group to an S3 Storage Lens dashboard.

Storage Lens groups permissions

Storage Lens groups require certain permissions in AWS Identity and Access Management (IAM) to authorize access to S3 Storage Lens group actions. To grant these permissions, you can use an identity-based IAM policy. You can attach this policy to IAM users, groups, or roles to grant them permissions. Such permissions can include the ability to create or delete Storage Lens groups, view their configurations, or manage their tags.

The IAM user or role that you grant permissions to must belong to the account that created or owns the Storage Lens group.

To use Storage Lens groups and to view your Storage Lens groups metrics, you must first have the appropriate permissions to use S3 Storage Lens. For more information, see the section called "Setting permissions".

To create and manage S3 Storage Lens groups, you must have the following IAM permissions, depending on which actions you want to perform:

Action	IAM permissions
Create a new Storage Lens group	`s3:CreateStorageLensGroup`
Create a new Storage Lens group with tags	`s3:CreateStorageLensGroup` , `s3:TagResource`
Update an existing Storage Lens group	`s3:UpdateStorageLensGroup`

Action	IAM permissions
Return the details of a Storage Lens group configuration	`s3:GetStorageLensGroup`
List all Storage Lens groups in your home Region	`s3:ListStorageLensGroups`
Delete a Storage Lens group	`s3:DeleteStorageLensGroup`
List the tags that were added to your Storage Lens group	`s3:ListTagsForResource`
Add or update a Storage Lens group tag for an existing Storage Lens group	`s3:TagResource`
Delete a tag from a Storage Lens group	`s3:UntagResource`

Here's an example of how to configure your IAM policy in the account that creates the Storage Lens group. To use this policy, replace *`us-east-1`* with the home Region that your Storage Lens group is located in. Replace *`111122223333`* with your AWS account ID, and replace *`example-storage-lens-group`* with the name of your Storage Lens group. To apply these permissions to all Storage Lens groups, replace *`example-storage-lens-group`* with an *`*`*.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "EXAMPLE-Statement-ID",
            "Effect": "Allow",
            "Action": [
                "s3:CreateStorageLensGroup",
                "s3:UpdateStorageLensGroup",
                "s3:GetStorageLensGroup",
                "s3:ListStorageLensGroups",
                "s3:DeleteStorageLensGroup",
                "s3:TagResource",
                "s3:UntagResource",
                "s3:ListTagsForResource"
                ],
```

```
            "Resource": "arn:aws:s3:us-east-1:111122223333:storage-lens-group/example-
storage-lens-group"
        }
    ]
}
```

For more information about S3 Storage Lens permissions, see Setting Amazon S3 Storage Lens permissions. For more information about IAM policy language, see Policies and permissions in Amazon S3.

Storage Lens groups configuration

S3 Storage Lens group name

We recommend giving your Storage Lens groups names that indicate their purpose so that you can easily determine which groups you want to attach to your dashboards. To attach a Storage Lens group to a dashboard, you must specify the group in the **Storage Lens group aggregation** section of the dashboard configuration.

Storage Lens group names must be unique within the account. They must not exceed 64 characters, and can contain only letters (a-z, A-Z), numbers (0-9), hyphens (-), and underscores (_).

Home Region

The home Region is the AWS Region where your Storage Lens group is created and maintained. Your Storage Lens group is created in the same home Region as your Amazon S3 Storage Lens dashboard. The Storage Lens group configuration and metrics are also stored in this Region. You can create up to 50 Storage Lens groups in a home Region.

After you create your Storage Lens group, you can't edit the home Region.

Scope

To include objects in your Storage Lens group, they must be in scope for your Amazon S3 Storage Lens dashboard. The scope of your Storage Lens dashboard is determined by the buckets that you included in the **Dashboard scope** of your S3 Storage Lens dashboard configuration.

You can use different filters for your objects to define the scope of your Storage Lens group. To view these Storage Lens group metrics in your S3 Storage Lens dashboard, objects must match the filters that you include in your Storage Lens groups. For example, suppose that your Storage Lens

group includes objects with the prefix `marketing` and the suffix `.png`, but no objects match those criteria. In this case, metrics for this Storage Lens group won't be generated in your daily metrics export, and no metrics for this group will be visible in your dashboard.

Filters

You can use the following filters in an S3 Storage Lens group:

- **Prefixes** – Specifies the prefix of included objects, which is a string of characters at the beginning of the object key name. For example, a value of `images` for the **Prefixes** filter includes objects with any of the following prefixes: `images/`, `images-marketing`, and `images/production`. The maximum length of a prefix is 1,024 bytes.
- **Suffixes** – Specifies the suffix of included objects (for example, `.png`, `.jpeg`, or `.csv`). The maximum length of a suffix is 1,024 bytes.
- **Object tags** – Specifies the list of object tags that you want to filter on. A tag key can't exceed 128 Unicode characters, and a tag value can't exceed 256 Unicode characters. Note that if the object tag value field is left empty, S3 Storage Lens groups only matches the object to other objects that also have empty tag values.
- **Age** – Specifies the object age range of included objects in days. Only integers are supported.
- **Size** – Specifies the object size range of included objects in bytes. Only integers are supported. The maximum allowable value is 50 TB.

Storage Lens group object tags

You can Creating a Storage Lens group that includes up to 10 object tag filters. The following example includes two object tag key-value pairs as filters for a Storage Lens group that's named *`Marketing-Department`*. To use this example, replace *`Marketing-Department`* with the name of your group, and replace *`object-tag-key-1`*, *`object-tag-value-1`*, and so on with the object tag key-value pairs that you want to filter on.

```
{
    "Name": "Marketing-Department",
    "Filter": {
     "MatchAnyTag":[
                {
                    "Key": "object-tag-key-1",
                    "Value": "object-tag-value-1"
                },
```

```
                {
                    "Key": "object-tag-key-2",
                    "Value": "object-tag-value-2"
                }
            ]
    }
}
```

Logical operators (And or Or)

To include multiple filter conditions in your Storage Lens group, you can use logical operators (either And or Or). In the following example, the Storage Lens group that's named *Marketing-Department* has an And operator that contains Prefix, ObjectAge, and ObjectSize filters. Because an And operator is used, only objects that match **all** of these filter conditions will be included the Storage Lens group's scope.

To use this example, replace the *user input placeholders* with the values that you want to filter on.

```
{
    "Name": "Marketing-Department",
    "Filter": {
        "And": {
            "MatchAnyPrefix": [
                "prefix-1",
                "prefix-2",
                "prefix-3/sub-prefix-1"
            ],
             "MatchObjectAge": {
                "DaysGreaterThan": 10,
                "DaysLessThan": 60
            },
            "MatchObjectSize": {
                "BytesGreaterThan": 10,
                "BytesLessThan": 60
            }
        }
    }
}
```

> **Note**
>
> If you want to include objects that match **any** of the conditions in the filters, replace the And logical operator with the `Or` logical operator in this example.

AWS resource tags

Each S3 Storage Lens group is counted as an AWS resource with its own Amazon Resource Name (ARN). Therefore, when you configure your Storage Lens group, you can optionally add AWS resource tags to the group. You can add up to 50 tags for each Storage Lens group. To create a Storage Lens group with tags, you must have the `s3:CreateStorageLensGroup` and `s3:TagResource` permissions.

You can use AWS resource tags to categorize resources according to department, line of business, or project. Doing so is useful when you have many resources of the same type. By applying tags, you can quickly identify a specific Storage Lens group based on the tags that you've assigned to it. You can also use tags to track and allocate costs.

In addition, when you add an AWS resource tag to your Storage Lens group, you activate attribute-based access control (ABAC). ABAC is an authorization strategy that defines permissions based on attributes, in this case tags. You can also use conditions that specify resource tags in your IAM policies to control access to AWS resources.

You can edit tag keys and values, and you can remove tags from a resource at any time. Also, be aware of the following limitations:

- Tag keys and tag values are case sensitive.
- If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value.
- If you delete a resource, any tags for the resource are also deleted.
- Don't include private or sensitive data in your AWS resource tags.
- System tags (or tags with tag keys that begin with `aws:`) aren't supported.
- The length of each tag key can't exceed 128 characters. The length of each tag value can't exceed 256 characters.

Storage Lens groups metrics export

S3 Storage Lens group metrics are included in the Amazon S3 Storage Lens metrics export for the dashboard that the Storage Lens group is attached to. For general information about the Storage Lens metrics export feature, see Viewing Amazon S3 Storage Lens metrics using a data export.

Your metrics export for Storage Lens groups includes any S3 Storage Lens metrics that are in scope for the dashboard that you attached the Storage Lens group to. The export also includes additional metrics data for Storage Lens groups.

After you create your Storage Lens group, your metrics export is sent daily to the bucket that you selected when you configured the metrics export for the dashboard that your group is attached to. It can take up to 48 hours for you to receive the first metrics export.

To generate metrics in the daily export, objects must match the filters that you include in your Storage Lens groups. If no objects match the filters that you included in your Storage Lens group, then no metrics will be generated. However, if an object matches two or more Storage Lens groups, the object is listed separately for each group when it appears in the metrics export.

You can identify metrics for Storage Lens groups by looking for one of the following values in the `record_type` column of the metrics export for your dashboard:

- `STORAGE_LENS_GROUP_BUCKET`
- `STORAGE_LENS_GROUP_ACCOUNT`

The `record_value` column displays the resource ARN for the Storage Lens group (for example, `arn:aws:s3:us-east-1:111122223333:storage-lens-group/Marketing-Department`).

Using Storage Lens groups

Amazon S3 Storage Lens groups aggregates metrics using custom filters based on object metadata. You can analyze and filter S3 Storage Lens metrics using prefixes, suffixes, object tags, object size, or object age. With Amazon S3 Storage Lens groups, you can also categorize your usage within and across Amazon S3 buckets. As a result, you'll be able to better understand and optimize your S3 storage.

To start visualizing the data for a Storage Lens group, you must first attach your Storage Lens group to an S3 Storage Lens dashboard. If you need to manage Storage Lens groups in the dashboard, you can edit the dashboard configuration. To check which Storage Lens groups are

under your account, you can list them. To check which Storage Lens groups are attached to your dashboard, you can always check the **Storage Lens groups** tab in the dashboard. To review or update the scope of an existing Storage Lens group, you can view its details. You can also permanently delete a Storage Lens group.

To manage permissions, you can create and add user-defined AWS resource tags to your Storage Lens groups. You can use AWS resource tags to categorize resources according to department, line of business, or project. Doing so is useful when you have many resources of the same type. By applying tags, you can quickly identify a specific Storage Lens group based on the tags that you've assigned to it.

In addition, when you add an AWS resource tag to your Storage Lens group, you activate attribute-based access control (ABAC). ABAC is an authorization strategy that defines permissions based on attributes, in this case tags. You can also use conditions that specify resource tags in your IAM policies to control access to AWS resources.

Topics

- Creating a Storage Lens group
- Attaching or removing S3 Storage Lens groups to or from your dashboard
- Visualizing your Storage Lens groups data
- Updating a Storage Lens group
- Managing AWS resource tags with Storage Lens groups
- Listing all Storage Lens groups
- Viewing Storage Lens group details
- Deleting a Storage Lens group

Creating a Storage Lens group

The following examples demonstrate how to create an Amazon S3 Storage Lens group by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To create a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region that you want to switch to.
3. In the left navigation pane, choose **Storage Lens groups**.
4. Choose **Create Storage Lens group**.
5. Under **General**, view your **Home Region** and enter your **Storage Lens group name**.
6. Under **Scope**, choose the filter that you want to apply to your Storage Lens group. To apply multiple filters, choose your filters, and then choose the **AND** or **OR** logical operator.

   - For the **Prefixes** filter, choose **Prefixes**, and enter a prefix string. To add multiple prefixes, choose **Add prefix**. To remove a prefix, choose **Remove** next to the prefix that you want to remove.
   - For the **Object tags** filter, choose **Object tags**, and enter the key-value pair for your object. Then, choose **Add tag**. To remove a tag, choose **Remove** next to the tag that you want to remove.
   - For the **Suffixes** filter, choose **Suffixes**, and enter a suffix string. To add multiple suffixes, choose **Add suffix**. To remove a suffix, choose **Remove** next to the suffix that you want to remove.
   - For the **Age** filter, specify the object age range in days. Choose **Specify minimum object age**, and enter the minimum object age. Then, choose **Specify maximum object age**, and enter the maximum object age.
   - For the **Size** filter, specify the object size range and unit of measurement. Choose **Specify minimum object size**, and enter the minimum object size. Choose **Specify maximum object size**, and enter the maximum object size.
7. (Optional) For AWS resource tags, add the key-value pair, and then choose **Add tag**.
8. Choose **Create Storage Lens group**.

Using the AWS CLI

The following example AWS CLI command creates a Storage Lens group. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control create-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

The following example AWS CLI command creates a Storage Lens group with two AWS resource tags. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control create-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json \
--tags Key=k1,Value=v1 Key=k2,Value=v2
```

For example JSON configurations, see Storage Lens groups configuration.

Using the AWS SDK for Java

The following AWS SDK for Java example creates a Storage Lens group. To use this example, replace the *user input placeholders* with your own information.

Example– Create a Storage Lens group with a single filter

The following example creates a Storage Lens group named *Marketing-Department*. This group has an object age filter that specifies the age range as *30* to *90* days. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;

public class CreateStorageLensGroupWithObjectAge {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            StorageLensGroupFilter objectAgeFilter = StorageLensGroupFilter.builder()
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
                            .daysLessThan(90)
```

```
                                .build())
                        .build();

                StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                        .name(storageLensGroupName)
                        .filter(objectAgeFilter)
                        .build();

                CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                        .storageLensGroup(storageLensGroup)
                        .accountId(accountId).build();

                S3ControlClient s3ControlClient = S3ControlClient.builder()
                        .region(Region.US_WEST_2)
                        .credentialsProvider(ProfileCredentialsProvider.create())
                        .build();
                s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
            } catch (AmazonServiceException e) {
                // The call was transmitted successfully, but Amazon S3 couldn't process
                // it and returned an error response.
                e.printStackTrace();
            } catch (SdkClientException e) {
                // Amazon S3 couldn't be contacted for a response, or the client
                // couldn't parse the response from Amazon S3.
                e.printStackTrace();
            }
        }
}
```

Example– Create a Storage Lens group with an AND operator that includes multiple filters

The following example creates a Storage Lens group named *Marketing-Department*. This group uses the AND operator to indicate that objects must match **all** of the filter conditions. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
```

```
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectAge;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.S3Tag;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupAndOperator;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;


public class CreateStorageLensGroupWithAndFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create object tags.
            S3Tag tag1 = S3Tag.builder()
                    .key("object-tag-key-1")
                    .value("object-tag-value-1")
                    .build();
            S3Tag tag2 = S3Tag.builder()
                    .key("object-tag-key-2")
                    .value("object-tag-value-2")
                    .build();

            StorageLensGroupAndOperator andOperator =
 StorageLensGroupAndOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .matchAnyTag(tag1, tag2)
                    .matchObjectAge(MatchObjectAge.builder()
                            .daysGreaterThan(30)
                            .daysLessThan(90).build())
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L).build())
                    .build();

            StorageLensGroupFilter andFilter = StorageLensGroupFilter.builder()
                    .and(andOperator)
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
```

```
                    .filter(andFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example– Create a Storage Lens group with an OR operator that includes multiple filters

The following example creates a Storage Lens group named *Marketing-Department*. This group uses an OR operator to apply a prefix filter (*prefix-1*, *prefix-2*, *prefix3/sub-prefix-1*) or an object size filter with a size range between *1000* bytes and *6000* bytes. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.MatchObjectSize;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
```

```
import software.amazon.awssdk.services.s3control.model.StorageLensGroupOrOperator;

public class CreateStorageLensGroupWithOrFilter {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            StorageLensGroupOrOperator orOperator =
 StorageLensGroupOrOperator.builder()
                    .matchAnyPrefix("prefix-1", "prefix-2", "prefix-3/sub-prefix-1")
                    .matchObjectSize(MatchObjectSize.builder()
                            .bytesGreaterThan(1000L)
                            .bytesLessThan(6000L)
                            .build())
                    .build();

            StorageLensGroupFilter orFilter = StorageLensGroupFilter.builder()
                    .or(orOperator)
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(orFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
```

```
        }
    }
}
```

Example– Create a Storage Lens group with a single filter and two AWS resource tags

The following example creates a Storage Lens group named *Marketing-Department* that has a suffix filter. This example also adds two AWS resource tags to the Storage Lens group. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.CreateStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.Tag;

public class CreateStorageLensGroupWithResourceTags {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            // Create AWS resource tags.
            Tag resourceTag1 = Tag.builder()
                    .key("resource-tag-key-1")
                    .value("resource-tag-value-1")
                    .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-value-2")
                    .build();

            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg")
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
```

```
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();

            CreateStorageLensGroupRequest createStorageLensGroupRequest =
 CreateStorageLensGroupRequest.builder()
                    .storageLensGroup(storageLensGroup)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId).build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.createStorageLensGroup(createStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

For example JSON configurations, see Storage Lens groups configuration.

Attaching or removing S3 Storage Lens groups to or from your dashboard

After you've upgraded to the advanced tier in Amazon S3 Storage Lens, you can attach a Storage Lens group to your dashboard. If you have several Storage Lens groups, you can include or exclude the groups that you want.

Your Storage Lens groups must reside within the designated home Region in the dashboard account. After you attach a Storage Lens group to your dashboard, you'll receive the additional Storage Lens group aggregation data in your metrics export within 48 hours.

> **Note**
>
> If you want to view aggregated metrics for your Storage Lens group, you must attach it to your Storage Lens dashboard. For examples of Storage Lens group JSON configuration files, see S3 Storage Lens example configuration with Storage Lens groups in JSON.

Using the S3 console

To attach a Storage Lens group to a Storage Lens dashboard

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, under **Storage Lens**, choose **Dashboards**.
3. Choose the option button for the Storage Lens dashboard that you want to attach a Storage Lens group to.
4. Choose **Edit**.
5. Under **Metrics selection**, choose **Advanced metrics and recommendations**.
6. Select **Storage Lens group aggregation**.

   > **Note**
   >
   > By default, **Advanced metrics** is also selected. However, you can also deselect this setting as it's not required to aggregate Storage Lens groups data.

7. Scroll down to **Storage Lens group aggregation** and specify the Storage Lens group or groups that you either want to include or exclude in the data aggregation. You can use the following filtering options:

   - If you want to include certain Storage Lens groups, choose **Include Storage Lens groups**. Under **Storage Lens groups to include**, select your Storage Lens groups.
   - If you want to include all Storage Lens groups, select **Include all Storage Lens groups in home Region in this account**.
   - If you want to exclude certain Storage Lens groups, choose **Exclude Storage Lens groups**. Under **Storage Lens groups to exclude**, select the Storage Lens groups that you want to exclude.

8. Choose **Save changes**. If you've configured your Storage Lens groups correctly, you will see the additional Storage Lens group aggregation data in your dashboard within 48 hours.

To remove a Storage Lens group from an S3 Storage Lens dashboard

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, under **Storage Lens**, choose **Dashboards**.
3. Choose the option button for the Storage Lens dashboard that you want to remove a Storage Lens group from.
4. Choose **View dashboard configuration**.
5. Choose **Edit**.
6. Scroll down to the **Metrics selection** section.
7. Under **Storage Lens group aggregation**, choose the **X** next to the Storage Lens group that you want to remove. This removes your Storage Lens group.

   If you included all of your Storage Lens groups in your dashboard, clear the check box next to **Include all Storage Lens groups in home Region in this account**.
8. Choose **Save changes**.

> **Note**
>
> It will take up to 48 hours for your dashboard to reflect the configuration updates.

Using the AWS SDK for Java

Example– Attach all Storage Lens groups to a dashboard

The following SDK for Java example attaches all Storage Lens groups in the account *111122223333* to the *DashBoardConfigurationId* dashboard:

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.services.s3control.AWSS3Control;
```

```
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWithStorageLensGroups {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel();

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
```

```
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example– Attach two Storage Lens groups to a dashboard

The following AWS SDK for Java example attaches two Storage Lens groups (*StorageLensGroupName1* and *StorageLensGroupName2*) to the *ExampleDashboardConfigurationId* dashboard.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroups {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
 StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName1,
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName2);
```

```
            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Example– Attach all Storage Lens groups with exclusions

The following SDK for Java example attaches all Storage Lens groups to the *ExampleDashboardConfigurationId* dashboard, excluding the two specified (*StorageLensGroupName1* and *StorageLensGroupName2*):

```
package aws.example.s3control;


import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3control.AWSS3Control;
import com.amazonaws.services.s3control.AWSS3ControlClient;
import com.amazonaws.services.s3control.model.AccountLevel;
import com.amazonaws.services.s3control.model.BucketLevel;
import com.amazonaws.services.s3control.model.PutStorageLensConfigurationRequest;
import com.amazonaws.services.s3control.model.StorageLensConfiguration;
import com.amazonaws.services.s3control.model.StorageLensGroupLevel;
import com.amazonaws.services.s3control.model.StorageLensGroupLevelSelectionCriteria;

import static com.amazonaws.regions.Regions.US_WEST_2;

public class CreateDashboardWith2StorageLensGroupsExcluded {
    public static void main(String[] args) {
        String configurationId = "ExampleDashboardConfigurationId";
        String storageLensGroupName1 = "StorageLensGroupName1";
        String storageLensGroupName2 = "StorageLensGroupName2";
        String sourceAccountId = "111122223333";

        try {
            StorageLensGroupLevelSelectionCriteria selectionCriteria = new
 StorageLensGroupLevelSelectionCriteria()
                    .withInclude(
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName1,
                            "arn:aws:s3:" + US_WEST_2.getName() + ":" + sourceAccountId
 + ":storage-lens-group/" + storageLensGroupName2);

            System.out.println(selectionCriteria);
            StorageLensGroupLevel storageLensGroupLevel = new StorageLensGroupLevel()
                    .withSelectionCriteria(selectionCriteria);

            AccountLevel accountLevel = new AccountLevel()
                    .withBucketLevel(new BucketLevel())
                    .withStorageLensGroupLevel(storageLensGroupLevel);

            StorageLensConfiguration configuration = new StorageLensConfiguration()
                    .withId(configurationId)
```

```
                    .withAccountLevel(accountLevel)
                    .withIsEnabled(true);

            AWSS3Control s3ControlClient = AWSS3ControlClient.builder()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(US_WEST_2)
                    .build();

            s3ControlClient.putStorageLensConfiguration(new
 PutStorageLensConfigurationRequest()
                    .withAccountId(sourceAccountId)
                    .withConfigId(configurationId)
                    .withStorageLensConfiguration(configuration)
            );
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Visualizing your Storage Lens groups data

You can visualize your Storage Lens groups data by attaching the group to your Amazon S3 Storage Lens dashboard. After you've included the Storage Lens group in the Storage Lens group aggregation in your dashboard configuration, it can take up to 48 hours for the Storage Lens group data to display in your dashboard.

After the dashboard configuration has been updated, any newly attached Storage Lens groups appear in the list of available resources under the **Storage Lens groups** tab. You can also further analyze storage usage in your **Overview** tab by slicing the data by another dimension. For example, you can choose one of the items listed under the **Top 3** categories and choose **Analyze by** to slice the data by another dimension. You can't apply the same dimension as the filter itself.

> **Note**
>
> You can't apply a Storage Lens group filter along with a prefix filter, or the reverse. You also can't further analyze a Storage Lens group by using a prefix filter.

You can use the **Storage Lens groups** tab in the Amazon S3 Storage Lens dashboard to customize the data visualization for the Storage Lens groups that are attached to your dashboard. You can either visualize the data for some Storage Lens groups that are attached to your dashboard, or all of them.

When visualizing Storage Lens group data in your S3 Storage Lens dashboard, be aware of the following:

- S3 Storage Lens aggregates usage metrics for an object under all matching Storage Lens groups. Therefore, if an object matches the filter conditions for two or more Storage Lens groups, you will see repeated counts for the same object across your storage usage.
- Objects must match the filters that you include in your Storage Lens groups. If no objects match the filters that you include in your Storage Lens group, then no metrics are generated. To determine if there are any unassigned objects, check your total object count in the dashboard at the account level and bucket level.

Updating a Storage Lens group

The following examples demonstrate how to update an Amazon S3 Storage Lens group. You can update a Storage Lens group by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To update a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the Storage Lens group that you want to update.
4. Under **Scope**, choose **Edit**.

5. On the **Scope** page, select the filter that you want to apply to your Storage Lens group. To apply multiple filters, select your filters, and choose the **AND** or **OR** logical operator.

   - For the **Prefixes** filter, select **Prefixes**, and enter a prefix string. To add multiple prefixes, choose **Add prefix**. To remove a prefix, choose **Remove** next to the prefix that you want to remove.
   - For the **Object tags** filter, enter the key-value pair for your object. Then, choose **Add tag**. To remove a tag, choose **Remove** next to the tag that you want to remove.
   - For the **Suffixes** filter, select **Suffixes**, and enter a suffix string. To add multiple suffixes, choose **Add suffix**. To remove a suffix, choose **Remove** next to the suffix that you want to remove.
   - For the **Age** filter, specify the object age range in days. Choose **Specify minimum object age**, and enter the minimum object age. For **Specify maximum object age**, enter the maximum object age.
   - For the **Size** filter, specify the object size range and unit of measurement. Choose **Specify minimum object size**, and enter the minimum object size. For **Specify maximum object size**, enter the maximum object size.

6. Choose **Save changes**. The details page for the Storage Lens group appears.
7. (Optional) If you want to add a new AWS resource tag, scroll to the **AWS resource tags** section, then choose **Add tags**. The **Add tags** page appears.

   Add the new key-value pair, then choose **Save changes**. The details page for the Storage Lens group appears.
8. (Optional) If you want to remove an existing AWS resource tag, scroll to the **AWS resource tags** section, and select the resource tag. Then, choose **Delete**. The **Delete AWS tags** dialog box appears.

   Choose **Delete** again to permanently delete the AWS resource tag.

   > **Note**
   >
   > After you permanently delete an AWS resource tag, it can't be restored.

Using the AWS CLI

The following AWS CLI example command returns the configuration details for a Storage Lens group named *marketing-department*. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

The following AWS CLI example updates a Storage Lens group. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control update-storage-lens-group --account-id 111122223333 \
--region us-east-1 --storage-lens-group=file://./marketing-department.json
```

For example JSON configurations, see Storage Lens groups configuration.

Using the AWS SDK for Java

The following AWS SDK for Java example returns the configuration details for the *Marketing-Department* Storage Lens group in account *111122223333*. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
```

```
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

The following example updates the Storage Lens group named *Marketing-Department* in account *111122223333*. This example updates the dashboard scope to include objects that match any of the following suffixes: *.png*, *.gif*, *.jpg*, or *.jpeg*. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.StorageLensGroup;
import software.amazon.awssdk.services.s3control.model.StorageLensGroupFilter;
import software.amazon.awssdk.services.s3control.model.UpdateStorageLensGroupRequest;

public class UpdateStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
```

```
            // Create updated filter.
            StorageLensGroupFilter suffixFilter = StorageLensGroupFilter.builder()
                    .matchAnySuffix(".png", ".gif", ".jpg", ".jpeg")
                    .build();

            StorageLensGroup storageLensGroup = StorageLensGroup.builder()
                    .name(storageLensGroupName)
                    .filter(suffixFilter)
                    .build();

            UpdateStorageLensGroupRequest updateStorageLensGroupRequest =
 UpdateStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .storageLensGroup(storageLensGroup)
                    .accountId(accountId)
                    .build();

            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.updateStorageLensGroup(updateStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

For example JSON configurations, see Storage Lens groups configuration.

Managing AWS resource tags with Storage Lens groups

Each Amazon S3 Storage Lens group is counted as an AWS resource with its own Amazon Resource Name (ARN). Therefore, when you configure your Storage Lens group, you can optionally add AWS resource tags to the group. You can add up to 50 tags for each Storage Lens group. To create a Storage Lens group with tags, you must have the `s3:CreateStorageLensGroup` and `s3:TagResource` permissions.

You can use AWS resource tags to categorize resources according to department, line of business, or project. Doing so is useful when you have many resources of the same type. By applying tags, you can quickly identify a specific Storage Lens group based on the tags that you've assigned to it. You can also use tags to track and allocate costs.

In addition, when you add an AWS resource tag to your Storage Lens group, you activate attribute-based access control (ABAC). ABAC is an authorization strategy that defines permissions based on attributes, in this case tags. You can also use conditions that specify resource tags in your IAM policies to control access to AWS resources.

You can edit tag keys and values, and you can remove tags from a resource at any time. Also, be aware of the following limitations:

- Tag keys and tag values are case sensitive.
- If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value.
- If you delete a resource, any tags for the resource are also deleted.
- Don't include private or sensitive data in your AWS resource tags.
- System tags (with tag keys that begin with `aws:`) aren't supported.
- The length of each tag key can't exceed 128 characters. The length of each tag value can't exceed 256 characters.

The following examples demonstrate how to use AWS resource tags with Storage Lens groups.

Topics

- Adding an AWS resource tag to a Storage Lens group
- Updating Storage Lens group tag values
- Deleting an AWS resource tag from a Storage Lens group
- Listing Storage Lens group tags

Adding an AWS resource tag to a Storage Lens group

The following examples demonstrate how to add AWS resource tags to an Amazon S3 Storage Lens group. You can add resource tags by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To add an AWS resource tag to a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the Storage Lens group that you want to update.
4. Under **AWS resource tags**, choose **Add tags**.
5. On the **Add tags** page, add the new key-value pair.

> **Note**
>
> Adding a new tag with the same key as an existing tag overwrites the previous tag value.

6. (Optional) To add more than one new tag, choose **Add tag** again to continue adding new entries. You can add up to 50 AWS resource tags to your Storage Lens group.
7. (Optional) If you want to remove a newly added entry, choose **Remove** next to the tag that you want to remove.
8. Choose **Save changes**.

Using the AWS CLI

The following example AWS CLI command adds two resource tags to an existing Storage Lens group named *marketing-department*. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-department \
--region us-east-1 --tags Key=k1,Value=v1 Key=k2,Value=v2
```

Using the AWS SDK for Java

The following AWS SDK for Java example adds two AWS resource tags to an existing Storage Lens group. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;

public class TagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag resourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-value-1")
                .build();
            Tag resourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(resourceTag1, resourceTag2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.tagResource(tagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
```

```
    }
}
```

Updating Storage Lens group tag values

The following examples demonstrate how to update Storage Lens group tag values by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To update an AWS resource tag for a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the Storage Lens group that you want to update.
4. Under **AWS resource tags**, select the tag that you want to update.
5. Add the new tag value, using the same key of the key-value pair that you want to update. Choose the checkmark icon to update the tag value.

   > **Note**
   >
   > Adding a new tag with the same key as an existing tag overwrites the previous tag value.

6. (Optional) If you want to add new tags, choose **Add tag** to add new entries. The **Add tags** page appears.

   You can add up to 50 AWS resource tags for your Storage Lens group. When you're finished adding new tags, choose **Save changes**.
7. (Optional) If you want to remove a newly added entry, choose **Remove** next to the tag that you want to remove. When you're finished removing tags, choose **Save changes**.

Using the AWS CLI

The following example AWS CLI command updates two tag values for the Storage Lens group named *`marketing-department`*. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control tag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
--region us-east-1 --tags Key=k1,Value=v3 Key=k2,Value=v4
```

Using the AWS SDK for Java

The following AWS SDK for Java example updates two Storage Lens group tag values. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;

public class UpdateTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            Tag updatedResourceTag1 = Tag.builder()
                .key("resource-tag-key-1")
                .value("resource-tag-updated-value-1")
                .build();
            Tag updatedResourceTag2 = Tag.builder()
                    .key("resource-tag-key-2")
                    .value("resource-tag-updated-value-2")
                    .build();
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tags(updatedResourceTag1, updatedResourceTag2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
```

```
            s3ControlClient.tagResource(tagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Deleting an AWS resource tag from a Storage Lens group

The following examples demonstrate how to delete an AWS resource tag from a Storage Lens group. You can delete tags by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To delete an AWS resource tag from a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the Storage Lens group that you want to update.
4. Under **AWS resource tags**, select the key-value pair that you want to delete.
5. Choose **Delete**. The **Delete AWS resource tags** dialog box appears.

   > **Note**
   >
   > If tags are used to control access, proceeding with this action can affect related resources. After you permanently delete a tag, it can't be restored.

6. Choose **Delete** to permanently delete the key-value pair.

Using the AWS CLI

The following AWS CLI command deletes two AWS resource tags from an existing Storage Lens group: To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control untag-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/Marketing-
Department \
--region us-east-1 --tag-keys k1 k2
```

Using the AWS SDK for Java

The following AWS SDK for Java example deletes two AWS resource tags from the Storage Lens group Amazon Resource Name (ARN) that you specify in account *111122223333*. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.UntagResourceRequest;

public class UntagResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            String tagKey1 = "resource-tag-key-1";
            String tagKey2 = "resource-tag-key-2";
            UntagResourceRequest untagResourceRequest = UntagResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .tagKeys(tagKey1, tagKey2)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
```

```
                    .build();
            s3ControlClient.untagResource(untagResourceRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Listing Storage Lens group tags

The following examples demonstrate how to list the AWS resource tags associated with a Storage Lens group. You can list tags by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To review the list of tags and tag values for a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the Storage Lens group that you're interested in.
4. Scroll down to the **AWS resource tags** section. All of the user-defined AWS resource tags that are added to your Storage Lens group are listed along with their tag values.

Using the AWS CLI

The following AWS CLI example command lists all the Storage Lens group tag values for the Storage Lens group named *marketing-department*. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control list-tags-for-resource --account-id 111122223333 \
--resource-arn arn:aws:s3:us-east-1:111122223333:storage-lens-group/marketing-
department \
```

```
--region us-east-1
```

Using the AWS SDK for Java

The following AWS SDK for Java example lists the Storage Lens group tag values for the Storage Lens group Amazon Resource Name (ARN) that you specify. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceRequest;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceResponse;

public class ListTagsForResource {
    public static void main(String[] args) {
        String resourceARN = "Resource_ARN";
        String accountId = "111122223333";

        try {
            ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                    .resourceArn(resourceARN)
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            ListTagsForResourceResponse response =
 s3ControlClient.listTagsForResource(listTagsForResourceRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
```

```
            e.printStackTrace();
        }
    }
}
```

Listing all Storage Lens groups

The following examples demonstrate how to list all Amazon S3 Storage Lens groups in an AWS account and home Region. These examples show how list all Storage Lens groups by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To list all Storage Lens groups in an account and home Region

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, the list of Storage Lens groups in your account is displayed.

Using the AWS CLI

The following AWS CLI example lists all of the Storage Lens groups for your account. To use this example command, replace the *`user input placeholders`* with your own information.

```
aws s3control list-storage-lens-groups --account-id 111122223333 \
--region us-east-1
```

Using the AWS SDK for Java

The following AWS SDK for Java example lists the Storage Lens groups for account *`111122223333`*. To use this example, replace the *`user input placeholders`* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
```

```
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsRequest;
import software.amazon.awssdk.services.s3control.model.ListStorageLensGroupsResponse;

public class ListStorageLensGroups {
    public static void main(String[] args) {
        String accountId = "111122223333";

        try {
            ListStorageLensGroupsRequest listStorageLensGroupsRequest =
 ListStorageLensGroupsRequest.builder()
                    .accountId(accountId)
                    .build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            ListStorageLensGroupsResponse response =
 s3ControlClient.listStorageLensGroups(listStorageLensGroupsRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Viewing Storage Lens group details

The following examples demonstrate how to view Amazon S3 Storage Lens group configuration details. You can view these details by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To view Storage Lens group configuration details

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the option button next to the Storage Lens group that you're interested in.
4. Choose **View details**. You can now review the details of your Storage Lens group.

Using the AWS CLI

The following AWS CLI example returns the configuration details for a Storage Lens group. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control get-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

Using the AWS SDK for Java

The following AWS SDK for Java example returns the configuration details for the Storage Lens group named *Marketing-Department* in account *111122223333*. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupRequest;
import software.amazon.awssdk.services.s3control.model.GetStorageLensGroupResponse;

public class GetStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            GetStorageLensGroupRequest getRequest =
 GetStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
```

```
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            GetStorageLensGroupResponse response =
 s3ControlClient.getStorageLensGroup(getRequest);
            System.out.println(response);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Deleting a Storage Lens group

The following examples demonstrate how to delete an Amazon S3 Storage Lens group by using the Amazon S3 console, AWS Command Line Interface (AWS CLI), and AWS SDK for Java.

Using the S3 console

To delete a Storage Lens group

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Storage Lens groups**.
3. Under **Storage Lens groups**, choose the option button next to the Storage Lens group that you want to delete.
4. Choose **Delete**. A **Delete Storage Lens group** dialog box displays.
5. Choose **Delete** again to permanently delete your Storage Lens group.

> **Note**
>
> After you delete a Storage Lens group, it can't be restored.

Using the AWS CLI

The following AWS CLI example deletes the Storage Lens group named *marketing-department*. To use this example command, replace the *user input placeholders* with your own information.

```
aws s3control delete-storage-lens-group --account-id 111122223333 \
--region us-east-1 --name marketing-department
```

Using the AWS SDK for Java

The following AWS SDK for Java example deletes the Storage Lens group named *Marketing-Department* in account *111122223333*. To use this example, replace the *user input placeholders* with your own information.

```
package aws.example.s3control;

import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.DeleteStorageLensGroupRequest;

public class DeleteStorageLensGroup {
    public static void main(String[] args) {
        String storageLensGroupName = "Marketing-Department";
        String accountId = "111122223333";

        try {
            DeleteStorageLensGroupRequest deleteStorageLensGroupRequest =
 DeleteStorageLensGroupRequest.builder()
                    .name(storageLensGroupName)
                    .accountId(accountId).build();
            S3ControlClient s3ControlClient = S3ControlClient.builder()
                    .region(Region.US_WEST_2)
                    .credentialsProvider(ProfileCredentialsProvider.create())
                    .build();
            s3ControlClient.deleteStorageLensGroup(deleteStorageLensGroupRequest);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it and returned an error response.
```

```
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Cataloging and analyzing your data with S3 Inventory

You can use Amazon S3 Inventory to help manage your storage. For example, you can use it to audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs. You can also simplify and speed up business workflows and big data jobs by using Amazon S3 Inventory, which provides a scheduled alternative to the Amazon S3 synchronous `List` API operations. Amazon S3 Inventory does not use the `List` API operations to audit your objects and does not affect the request rate of your bucket.

Amazon S3 Inventory provides comma-separated values (CSV), Apache optimized row columnar (ORC) or Apache Parquet output files that list your objects and their corresponding metadata on a daily or weekly basis for an S3 bucket or objects with a shared prefix (that is, objects that have names that begin with a common string). If you set up a weekly inventory, a report is generated every Sunday (UTC time zone) after the initial report. For information about Amazon S3 Inventory pricing, see Amazon S3 pricing.

You can configure multiple inventory lists for a bucket. When you're configuring an inventory list, you can specify the following:

- What object metadata to include in the inventory
- Whether to list all object versions or only current versions
- Where to store the inventory list file output
- Whether to generate the inventory on a daily or weekly basis
- Whether to encrypt the inventory list file

You can query Amazon S3 Inventory with standard SQL queries by using Amazon Athena, Amazon Redshift Spectrum, and other tools, such as Presto, Apache Hive, and Apache Spark. For more information about using Athena to query your inventory files, see the section called “Querying inventory with Athena”.

> **Note**
>
> It might take up to 48 hours for Amazon S3 to deliver the first inventory report.

> **Note**
>
> After deleting an inventory configuration, Amazon S3 might still deliver one additional inventory report during a brief transition period while the system processes the deletion.

Source and destination buckets

The bucket that the inventory lists objects for is called the *source bucket*. The bucket where the inventory list file is stored is called the *destination bucket*.

Source bucket

The inventory lists the objects that are stored in the source bucket. You can get an inventory list for an entire bucket, or you can filter the list by object key name prefix.

The source bucket:

- Contains the objects that are listed in the inventory
- Contains the configuration for the inventory

Destination bucket

Amazon S3 Inventory list files are written to the destination bucket. To group all the inventory list files in a common location in the destination bucket, you can specify a destination prefix in the inventory configuration.

The destination bucket:

- Contains the inventory file lists.
- Contains the manifest files that list all the inventory list files that are stored in the destination bucket. For more information, see Inventory manifest.
- Must have a bucket policy to give Amazon S3 permission to verify ownership of the bucket and permission to write files to the bucket.

- Must be in the same AWS Region as the source bucket.
- Can be the same as the source bucket.
- Can be owned by a different AWS account than the account that owns the source bucket.

Amazon S3 Inventory list

An inventory list file contains a list of the objects in the source bucket and metadata for each object. An inventory list file is stored in the destination bucket with one of the following formats:

- As a CSV file compressed with GZIP
- As an Apache optimized row columnar (ORC) file compressed with ZLIB
- As an Apache Parquet file compressed with Snappy

> **Note**
>
> Objects in Amazon S3 Inventory reports aren't guaranteed to be sorted in any order.

An inventory list file contains a list of the objects in the source bucket and metadata for each listed object. The following fields are always included:

- **Bucket name** – The name of the bucket that the inventory is for.
- **Key name** – The object key name (or key) that uniquely identifies the object in the bucket. When you're using the CSV file format, the key name is URL-encoded and must be decoded before you can use it.

You can choose to include the following additional metadata fields in the report:

- **ETag** – The entity tag (ETag) is a hash of the object. The ETag reflects changes only to the contents of an object, not to its metadata. The ETag can be an MD5 digest of the object data. Whether it is depends on how the object was created and how it is encrypted. For more information, see Object in the *Amazon Simple Storage Service API Reference*.
- **Last modified date** – The object creation date or the last modified date, whichever is the latest.
- **Size** – The object size in bytes, not including the size of incomplete multipart uploads, object metadata, and delete markers.

- **Storage class** – The storage class that's used for storing the object. Set to `STANDARD`, `REDUCED_REDUNDANCY`, `STANDARD_IA`, `ONEZONE_IA`, `INTELLIGENT_TIERING`, `GLACIER`, `DEEP_ARCHIVE`, `OUTPOSTS`, `GLACIER_IR`, `SNOW`, or `EXPRESS_ONEZONE`. For more information, see Understanding and managing Amazon S3 storage classes.
- **Checksum algorithm** – Indicates the algorithm that's used to create the checksum for the object. For more information, see Using supported checksum algorithms.
- **Encryption status** – The server-side encryption status, depending on what kind of encryption key is used— server-side encryption with Amazon S3 managed keys (SSE-S3), server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), dual-layer server-side encryption with AWS KMS keys (DSSE-KMS), or server-side encryption with customer-provided keys (SSE-C). Set to `SSE-S3`, `SSE-KMS`, `DSSE-KMS`, `SSE-C`, or `NOT-SSE`. A status of `NOT-SSE` means that the object is not encrypted with server-side encryption. For more information, see Protecting data with encryption.
- **S3 Intelligent-Tiering access tier** – Access tier (frequent or infrequent) of the object if it is stored in the S3 Intelligent-Tiering storage class. Set to `FREQUENT`, `INFREQUENT`, `ARCHIVE_INSTANT_ACCESS`, `ARCHIVE`, or `DEEP_ARCHIVE`. For more information, see Storage class for automatically optimizing data with changing or unknown access patterns.
- **S3 Intelligent-Tiering reference date** – The date that S3 Intelligent-Tiering uses to determine tier-transition eligibility for an object in the S3 Intelligent-Tiering storage class. This date reflects when the object was last considered active for tiering purposes. Amazon S3 sets this date from the object's creation date for objects created directly in S3 Intelligent-Tiering and never accessed, its transition date for objects transitioned into S3 Intelligent-Tiering and never accessed, or its most recent access date for objects accessed by a `GET`, `PUT`, `RESTORE`, or `SELECT` request. This field is empty for objects that aren't stored in the S3 Intelligent-Tiering storage class, and for objects smaller than 128 KB, which S3 Intelligent-Tiering doesn't monitor. For more information, see Storage class for automatically optimizing data with changing or unknown access patterns.

> **Note**
>
> The S3 Intelligent-Tiering reference date tracks object access on a best-effort basis and might not capture every access event, similar to Amazon S3 server access logs. As a result, the reference date might not reflect the most recent access to an object.

> **Note**
>
> A SELECT (SelectObjectContent) request updates the reference date and keeps an object out of the asynchronous Archive Access and Deep Archive Access tiers, but unlike a GET it doesn't move the object back to the Frequent Access tier or stop it tiering down to Infrequent Access or Archive Instant Access. An object accessed recently by SELECT can therefore show a recent reference date while remaining in Infrequent Access or Archive Instant Access. For more information, see How S3 Intelligent-Tiering works.

- **S3 Object Lock retain until date** – The date until which the locked object cannot be deleted. For objects with an active event hold, this date is dynamic, based on the report generation time. For objects with a released event hold or fixed retention, this date is fixed. For more information, see Locking objects with Object Lock.
- **S3 Object Lock retention mode** – Set to GOVERNANCE or COMPLIANCE for objects that are locked. For more information, see Locking objects with Object Lock.
- **S3 Object Lock legal hold status** – Set to ON if a legal hold has been applied to an object. Otherwise, it is set to OFF. For more information, see Locking objects with Object Lock.
- **S3 Object Lock event hold status** – Set to ON or OFF, reflecting the event hold status of the object. This field is empty when the object has no event hold configured.
- **S3 Object Lock event hold duration** – The variable retention duration that is configured for the object. This field is empty when the object has no event hold configured.

> **Note**
>
> When you include the S3 Object Lock event hold duration field, the report also includes an S3 Object Lock event hold duration unit column (DAYS or YEARS) to express the unit of the duration value.

- **Version ID** – The object version ID. When you enable versioning on a bucket, Amazon S3 assigns a version number to objects that are added to the bucket. For more information, see Retaining multiple versions of objects with S3 Versioning. (This field is not included if the list is configured only for the current version of the objects.)
- **IsLatest** – Set to True if the object is the current version of the object. (This field is not included if the list is configured only for the current version of the objects.)

- **Delete marker** – Set to `True` if the object is a delete marker. For more information, see Retaining multiple versions of objects with S3 Versioning. (This field is automatically added to your report if you've configured the report to include all versions of your objects).
- **Multipart upload flag** – Set to `True` if the object was uploaded as a multipart upload. For more information, see Uploading and copying objects using multipart upload in Amazon S3.
- **Object owner** – The canonical user ID of the owner of the object. For more information, see Find the canonical user ID for your AWS account in the *AWS Account Management Reference Guide*.
- **Replication status** – Set to `PENDING`, `COMPLETED`, `FAILED`, or `REPLICA`. For more information, see Getting replication status information.
- **S3 Bucket Key status** – Set to `ENABLED` or `DISABLED`. Indicates whether the object uses an S3 Bucket Key for SSE-KMS. For more information, see Using Amazon S3 Bucket Keys.
- **Object access control list** – An access control list (ACL) for each object that defines which AWS accounts or groups are granted access to this object and the type of access that is granted. The Object ACL field is defined in JSON format. An S3 Inventory report includes ACLs that are associated with objects in your source bucket, even when ACLs are disabled for the bucket. For more information, see Working with the Object ACL field and Access control list (ACL) overview.

Note

The Object ACL field is defined in JSON format. An inventory report displays the value for the Object ACL field as a base64-encoded string.
For example, suppose that you have the following Object ACL field in JSON format:

```
{
        "version": "2022-11-10",
        "status": "AVAILABLE",
        "grants": [{
            "canonicalId": "example-canonical-user-ID",
            "type": "CanonicalUser",
            "permission": "READ"
        }]
}
```

The Object ACL field is encoded and shown as the following base64-encoded string:

```
eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2Fs
```

To get the decoded value in JSON for the Object ACL field, you can query this field in Amazon Athena. For query examples, see Querying Amazon S3 Inventory with Amazon Athena.

- **Lifecycle Expiration Date** – Set to the lifecycle expiration timestamp of the object. This field will only be populated, if the object is to be expired by an applicable lifecycle rule. In other cases, the field will be empty. Objects with `FAILED` replication status will not have an expiration date populated, as S3 Lifecycle prevents expiration and transition actions on these objects until replication has succeeded. For more information, see Expiring objects.

Note

When an object reaches the end of its lifetime based on its lifecycle configuration, Amazon S3 queues the object for removal and removes it asynchronously. Therefore, there might be a delay between the expiration date and the date when Amazon S3 removes an object. The inventory report includes the objects that have expired but haven't been removed yet. For more information about expiration actions in S3 Lifecycle, see Expiring objects.

The following is an example inventory report with additional metadata fields consisting of four records.

```
amzn-s3-demo-bucket1    example-object-1    EXAMPLEDC8l.XJCENlF7LePaNIIvs001    TRUE
     1500    2024-08-15T15:28:26.0004    EXAMPLE21e1518b92f3d92773570f600    STANDARD
  FALSE    COMPLETED    SSE-KMS    2025-01-25T15:28:26.000Z    COMPLIANCE    Off
 ENABLED
 eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3sicGVybWlzc2lvbiI6Ik
    EXAMPLE766e8f6b115d93d41df2eac420aa4a465d177c1398bc6a088c76b7000
amzn-s3-demo-bucket1    example-object-2    EXAMPLEDC8l.XJCENlF7LePaNIIvs002
 TRUE        200    2024-08-21T15:28:26.000Z    EXAMPLE21e1518b92f3d92773570f601
  INTELLIGENT_TIERING    FALSE    COMPLETED    SSE-KMS    2025-01-25T15:28:26.000Z
    COMPLIANCE    Off    INFREQUENT    ENABLED    SHA-256
 eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3sicGVybWlzc2lvbiI6Ik
    EXAMPLE766e8f6b115d93d41df2eac420aa4a465d177c1398bc6a088c76b7001
amzn-s3-demo-bucket1    example-object-3    EXAMPLEDC8l.XJCENlF7LePaNIIvs003    TRUE
      12500    2023-01-15T15:28:30.000Z    EXAMPLE21e1518b92f3d92773570f602    STANDARD
    FALSE    REPLICA    SSE-KMS    2025-01-25T15:28:26.000Z    GOVERNANCE    On
 ENABLED
```

```
 eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3sicGVybWlzc2lvbiI6Ik
    EXAMPLE766e8f6b115d93d41df2eac420aa4a465d177c1398bc6a088c76b7002
amzn-s3-demo-bucket1    example-object-4    EXAMPLEDC8l.XJCENlF7LePaNIIvs004    TRUE
      100    2021-02-15T15:28:27.000Z    EXAMPLE21e1518b92f3d92773570f603    STANDARD
  FALSE    COMPLETED    SSE-KMS    2025-01-25T15:28:26.000Z    COMPLIANCE    Off
 ENABLED
 eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3sicGVybWlzc2lvbiI6Ik
    EXAMPLE766e8f6b115d93d41df2eac420aa4a465d177c1398bc6a088c76b7003
```

We recommend that you create a lifecycle policy that deletes old inventory lists. For more information, see Managing the lifecycle of objects.

The `s3:PutInventoryConfiguration` permission allows a user to both select all the metadata fields that are listed earlier for each object when configuring an inventory list and to specify the destination bucket to store the inventory. A user with read access to objects in the destination bucket can access all object metadata fields that are available in the inventory list. To restrict access to an inventory report, see Grant permissions for S3 Inventory and S3 analytics.

Inventory consistency

All of your objects might not appear in each inventory list. The inventory list provides eventual consistency for PUT requests (of both new objects and overwrites) and for DELETE requests. Each inventory list for a bucket is a snapshot of bucket items. These lists are eventually consistent (that is, a list might not include recently added or deleted objects).

To validate the state of an object before you take action on the object, we recommend that you perform a `HeadObject` REST API request to retrieve metadata for the object, or to check the object's properties in the Amazon S3 console. You can also check object metadata with the AWS CLI or the AWS SDKS. For more information, see `HeadObject` in the *Amazon Simple Storage Service API Reference*.

For more information about working with Amazon S3 Inventory, see the following topics.

Topics

- Configuring Amazon S3 Inventory
- Locating your inventory list
- Setting up Amazon S3 Event Notifications for inventory completion
- Querying Amazon S3 Inventory with Amazon Athena
- Converting empty version ID strings in Amazon S3 Inventory reports to null strings

- Working with the Object ACL field

Configuring Amazon S3 Inventory

Amazon S3 Inventory provides a flat file list of your objects and metadata, on a schedule that you define. You can use S3 Inventory as a scheduled alternative to the Amazon S3 synchronous `List` API operation. S3 Inventory provides comma-separated values (CSV), Apache optimized row columnar (ORC), or Apache Parquet (Parquet) output files that list your objects and their corresponding metadata.

You can configure S3 Inventory to create inventory lists on a daily or weekly basis for an S3 bucket or for objects that share a prefix (objects that have names that begin with the same string). For more information, see Cataloging and analyzing your data with S3 Inventory.

This section describes how to configure an inventory, including details about the inventory source and destination buckets.

Topics

- Overview
- Creating a destination bucket policy
- Granting Amazon S3 permission to use your customer managed key for encryption
- Configuring inventory by using the S3 console
- Using the REST API to work with S3 Inventory

Overview

Amazon S3 Inventory helps you manage your storage by creating lists of the objects in an S3 bucket on a defined schedule. You can configure multiple inventory lists for a bucket. The inventory lists are published to CSV, ORC, or Parquet files in a destination bucket.

The easiest way to set up an inventory is by using the Amazon S3 console, but you can also use the Amazon S3 REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs. The console performs the first step of the following procedure for you: adding a bucket policy to the destination bucket.

To set up Amazon S3 Inventory for an S3 bucket

1. **Add a bucket policy for the destination bucket.**

You must create a bucket policy on the destination bucket that grants permissions to Amazon S3 to write objects to the bucket in the defined location. For an example policy, see Grant permissions for S3 Inventory and S3 analytics.

2. **Configure an inventory to list the objects in a source bucket and publish the list to a destination bucket.**

   When you configure an inventory list for a source bucket, you specify the destination bucket where you want the list to be stored, and whether you want to generate the list daily or weekly. You can also configure whether to list all object versions or only current versions and what object metadata to include.

   Some object metadata fields in S3 Inventory report configurations are optional, meaning that they're available by default but they can be restricted when you grant a user the `s3:PutInventoryConfiguration` permission. You can control whether users can include these optional metadata fields in their reports by using the `s3:InventoryAccessibleOptionalFields` condition key.

   For more information about the optional metadata fields available in S3 Inventory, see OptionalFields in the *Amazon Simple Storage Service API Reference*. For more information about restricting access to certain optional metadata fields in an inventory configuration, see Control S3 Inventory report configuration creation.

   You can specify that the inventory list file be encrypted by using server-side encryption with an Amazon S3 managed key (SSE-S3) or an AWS Key Management Service (AWS KMS) customer managed key (SSE-KMS).

   > **Note**
   >
   > The AWS managed key (`aws/s3`) is not supported for SSE-KMS encryption with S3 Inventory.

   For more information about SSE-S3 and SSE-KMS, see Protecting data with server-side encryption. If you plan to use SSE-KMS encryption, see Step 3.

   - For information about how to use the console to configure an inventory list, see Configuring inventory by using the S3 console.

- To use the Amazon S3 API to configure an inventory list, use the PutBucketInventoryConfiguration REST API operation or the equivalent from the AWS CLI or AWS SDKs.

3. **To encrypt the inventory list file with SSE-KMS, grant Amazon S3 permission to use the AWS KMS key.**

   You can configure encryption for the inventory list file by using the Amazon S3 console, Amazon S3 REST API, AWS CLI, or AWS SDKs. Whichever way you choose, you must grant Amazon S3 permission to use the customer managed key to encrypt the inventory file. You grant Amazon S3 permission by modifying the key policy for the customer managed key that you want to use to encrypt the inventory file. Make sure that you've provided a KMS key ARN in the S3 Inventory configuration or the destination bucket's encryption settings. If no KMS key ARN has been specified and the default encryption settings are being used, you won't be able to access your S3 Inventory report.

   The destination bucket that stores the inventory list file can be owned by a different AWS account than the account that owns the source bucket. If you use SSE-KMS encryption for the cross-account operations of Amazon S3 Inventory, we recommend that you use a fully qualified KMS key ARN when you configure S3 inventory. For more information, see Using SSE-KMS encryption for cross-account operations and ServerSideEncryptionByDefault in the *Amazon Simple Storage Service API Reference*.

   > **Note**
   >
   > If you can't access your S3 Inventory report, use the GetBucketEncryption API, and check whether the destination bucket has the default SSE-KMS encryption enabled. If no KMS key ARN has been specified and the default encryption settings are being used, you won't be able to access your S3 Inventory report. To access S3 Inventory reports again, either provide a KMS key ARN in the S3 Inventory configuration or in the destination bucket's encryption settings.

> **Directory buckets**
>
> S3 Inventory is supported for directory buckets. When configuring S3 Inventory for a directory bucket, note the following differences:

- **Permissions** – For directory buckets, you must use the `s3express:PutInventoryConfiguration` and `s3express:GetInventoryConfiguration` permissions in an IAM identity-based policy instead of a bucket policy. These permissions use the `s3express:` namespace rather than the `s3:` namespace used for general purpose buckets. For more information, see Authorizing Regional endpoint API operations with IAM.
- **Supported optional fields** – The following optional fields are supported for directory buckets: `Size`, `LastModifiedDate`, `StorageClass`, `ETag`, `IsMultipartUploaded`, `EncryptionStatus`, `BucketKeyStatus`, `ChecksumAlgorithm`, and `LifecycleExpirationDate`.
- **Condition key** – For directory buckets, use the `s3express:InventoryAccessibleOptionalFields` condition key to control access to optional metadata fields in inventory reports.

Creating a destination bucket policy

If you create your inventory configuration through the Amazon S3 console, Amazon S3 automatically creates a bucket policy on the destination bucket that grants Amazon S3 write permission to the bucket. However, if you create your inventory configuration through the AWS CLI, AWS SDKs, or the Amazon S3 REST API, you must manually add a bucket policy on the destination bucket. The S3 Inventory destination bucket policy allows Amazon S3 to write data for the inventory reports to the bucket.

The following is the example bucket policy.

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
            "Sid": "InventoryExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3.amazonaws.com"
            },
```

```
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-DESTINATION-BUCKET/*"
            ],
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:s3:::DOC-EXAMPLE-SOURCE-BUCKET"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

For more information, see Grant permissions for S3 Inventory and S3 analytics.

Directory buckets

For directory buckets, you must manually add a destination bucket policy. The destination bucket policy uses a different service principal and ARN format than general purpose buckets. Specify `s3express.amazonaws.com` as the service principal, and use the directory bucket ARN format for the source bucket. The following example shows a destination bucket policy for directory buckets.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "InventoryExamplePolicy",
            "Effect": "Allow",
            "Principal": {
                "Service": "s3express.amazonaws.com"
            },
            "Action": "s3:PutObject",
            "Resource": [
                "arn:aws:s3:::DOC-EXAMPLE-DESTINATION-BUCKET/*"
            ],
            "Condition": {
```

```
                "ArnLike": {
                    "aws:SourceARN": "arn:aws:s3express:us-
west-2:111122223333:bucket/DOC-EXAMPLE-SOURCE-BUCKET--zone-id--x-s3"
                },
                "StringEquals": {
                    "aws:SourceAccount": "111122223333",
                    "s3:x-amz-acl": "bucket-owner-full-control"
                }
            }
        }
    ]
}
```

If an error occurs when you try to create the bucket policy, you are given instructions on how to fix it. For example, if you choose a destination bucket in another AWS account and don't have permissions to read and write to the bucket policy, you see an error message.

In this case, the destination bucket owner must add the bucket policy to the destination bucket. If the policy is not added to the destination bucket, you won't get an inventory report because Amazon S3 doesn't have permission to write to the destination bucket. If the source bucket is owned by a different account than that of the current user, the correct account ID of the source bucket owner must be substituted in the policy.

Note

Ensure that there are no Deny statements added to the destination bucket policy that would prevent the delivery of inventory reports into this bucket. For more information, see Why can't I generate an Amazon S3 Inventory Report? .

Granting Amazon S3 permission to use your customer managed key for encryption

To grant Amazon S3 permission to use your AWS Key Management Service (AWS KMS) customer managed key for server-side encryption, you must use a key policy. To update your key policy so that you can use your customer managed key, use the following procedure.

To grant Amazon S3 permissions to encrypt by using your customer managed key

1. Using the AWS account that owns the customer managed key, sign into the AWS Management Console.
2. Open the AWS KMS console at https://console.aws.amazon.com/kms.
3. To change the AWS Region, use the Region selector in the upper-right corner of the page.
4. In the left navigation pane, choose **Customer managed keys**.
5. Under **Customer managed keys**, choose the customer managed key that you want to use to encrypt your inventory files.
6. In the **Key policy** section, choose **Switch to policy view**.
7. To update the key policy, choose **Edit**.
8. On the **Edit key policy** page, add the following lines to the existing key policy. For *source-account-id* and *amzn-s3-demo-source-bucket*, supply the appropriate values for your use case.

```
{
    "Sid": "Allow Amazon S3 use of the customer managed key",
    "Effect": "Allow",
    "Principal": {
        "Service": "s3.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition":{
      "StringEquals":{
         "aws:SourceAccount":"source-account-id"
     },
      "ArnLike":{
        "aws:SourceARN": "arn:aws:s3:::amzn-s3-demo-source-bucket"
     }
   }
}
```

9. Choose **Save changes**.

For more information about creating customer managed keys and using key policies, see the following links in the *AWS Key Management Service Developer Guide*:

- Managing keys
- Key policies in AWS KMS

Directory buckets

For directory buckets, the KMS key policy uses a different service principal and source ARN format than general purpose buckets. Specify `s3express.amazonaws.com` as the service principal, and use the directory bucket ARN format for the source ARN. The following example shows the key policy statement for directory buckets.

```
{
    "Sid": "Allow S3 Express use of the KMS key",
    "Effect": "Allow",
    "Principal": {
        "Service": "s3express.amazonaws.com"
    },
    "Action": [
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "aws:SourceAccount": "source-account-id"
        },
        "ArnLike": {
            "aws:SourceARN": "arn:aws:s3express:region:source-account-
id:bucket/DOC-EXAMPLE-SOURCE-BUCKET--zone-id--x-s3"
        }
    }
}
```

Note

Ensure that there are no Deny statements added to the destination bucket policy that would prevent the delivery of inventory reports into this bucket. For more information, see Why can't I generate an Amazon S3 Inventory Report? .

Configuring inventory by using the S3 console

Use these instructions to configure inventory by using the S3 console.

> **Note**
>
> It might take up to 48 hours for Amazon S3 to deliver the first inventory report.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.

   > **Note**
   >
   > Configuring S3 Inventory for directory buckets is not supported in the Amazon Simple Storage Service console. To configure S3 Inventory for directory buckets, use the Amazon S3 REST API, AWS Command Line Interface (AWS CLI), or AWS SDKs.

3. In the buckets list, choose the name of the bucket that you want to configure Amazon S3 Inventory for.
4. Choose the **Management** tab.
5. Under **Inventory configurations**, choose **Create inventory configuration**.
6. For **Inventory configuration name**, enter a name.
7. For **Inventory scope**, do the following:
   - Enter an optional prefix. Don't include the bucket name in the prefix.
   - Choose which object versions to include, either **Current versions only** or **Include all versions**.
8. Under **Report details**, choose the location of the AWS account that you want to save the reports to: **This account** or **A different account**.
9. Under **Destination**, choose the destination bucket where you want the inventory reports to be saved.

   The destination bucket must be in the same AWS Region as the bucket for which you are setting up the inventory. The destination bucket can be in a different AWS account. When

specifying the destination bucket, you can also include an optional prefix to group your inventory reports together.

Under the **Destination** bucket field, you see the **Destination bucket permission** statement that is added to the destination bucket policy to allow Amazon S3 to place data in that bucket. For more information, see Creating a destination bucket policy.

10. Under **Frequency**, choose how often the report will be generated, **Daily** or **Weekly**.
11. For **Output format**, choose one of the following formats for the report:
    - **CSV** – If you plan to use this inventory report with S3 Batch Operations or if you want to analyze this report in another tool, such as Microsoft Excel, choose **CSV**.
    - **Apache ORC**
    - **Apache Parquet**
12. Under **Status**, choose **Enable** or **Disable**.
13. To configure server-side encryption, under **Inventory report encryption**, follow these steps:
    a. Under **Server-side encryption**, choose either **Do not specify an encryption key** or **Specify an encryption key** to encrypt data.
       - To keep the bucket settings for default server-side encryption of objects when storing them in Amazon S3, choose **Do not specify an encryption key**. As long as the bucket destination has S3 Bucket Keys enabled, the copy operation applies an S3 Bucket Key at the destination bucket.

         Note

         If the bucket policy for the specified destination requires objects to be encrypted before storing them in Amazon S3, you must choose **Specify an encryption key**. Otherwise, copying objects to the destination will fail.

       - To encrypt objects before storing them in Amazon S3, choose **Specify an encryption key**.

    b. If you chose **Specify an encryption key**, under **Encryption type**, you must choose either **Amazon S3 managed key (SSE-S3)** or **AWS Key Management Service key (SSE-KMS)**.

       SSE-S3 uses one of the strongest block ciphers—256-bit Advanced Encryption Standard (AES-256) to encrypt each object. SSE-KMS provides you with more control over your

key. For more information about SSE-S3, see Using server-side encryption with Amazon S3 managed keys (SSE-S3). For more information about SSE-KMS, see Using server-side encryption with AWS KMS keys (SSE-KMS).

> **Note**
>
> To encrypt the inventory list file with SSE-KMS, you must grant Amazon S3 permission to use the customer managed key. For instructions, see Grant Amazon S3 Permission to Encrypt Using Your KMS Keys.

c. If you chose **AWS Key Management Service key (SSE-KMS)**, under **AWS KMS key**, you can specify your AWS KMS key through one of the following options.

> **Note**
>
> If the destination bucket that stores the inventory list file is owned by a different AWS account, make sure that you use a fully qualified KMS key ARN to specify your KMS key.

- To choose from a list of available KMS keys, choose **Choose from your AWS KMS keys**, and choose a symmetric encryption KMS key from the list of available keys. Make sure the KMS key is in the same Region as your bucket.

  > **Note**
  >
  > Both the AWS managed key (`aws/s3`) and your customer managed keys appear in the list. However, the AWS managed key (`aws/s3`) is not supported for SSE-KMS encryption with S3 Inventory.

- To enter the KMS key ARN, choose **Enter AWS KMS key ARN**, and enter your KMS key ARN in the field that appears.
- To create a new customer managed key in the AWS KMS console, choose **Create a KMS key**.

14. For **Additional metadata fields**, select one or more of the following to add to the inventory report:

- **Size** – The object size in bytes, not including the size of incomplete multipart uploads, object metadata, and delete markers.
- **Last modified date** – The object creation date or the last modified date, whichever is the latest.
- **Multipart upload** – Specifies that the object was uploaded as a multipart upload. For more information, see Uploading and copying objects using multipart upload in Amazon S3.
- **Replication status** – The replication status of the object. For more information, see Getting replication status information.
- **Encryption status** – The server-side encryption type that's used to encrypt the object. For more information, see Protecting data with server-side encryption.
- **Bucket Key status** – Indicates whether a bucket-level key generated by AWS KMS applies to the object. For more information, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.
- **Object access control list** – An access control list (ACL) for each object that defines which AWS accounts or groups are granted access to this object and the type of access that is granted. For more information about this field, see Working with the Object ACL field. For more information about ACLs, see Access control list (ACL) overview.
- **Object owner** – The owner of the object.
- **Storage class** – The storage class that's used for storing the object.
- **Intelligent-Tiering: Access tier** – Indicates the access tier (frequent or infrequent) of the object if it was stored in the S3 Intelligent-Tiering storage class. For more information, see Storage class for automatically optimizing data with changing or unknown access patterns.
- You can add the optional **S3 Intelligent-Tiering reference date** field to your inventory report. For objects in the S3 Intelligent-Tiering storage class, this field reports the date that S3 Intelligent-Tiering uses to determine tier-transition eligibility. This date reflects when the object was last considered active for tiering. To include this field, select **S3 Intelligent-Tiering reference date** under the additional metadata fields when you create or edit an inventory configuration, or add `IntelligentTieringReferenceDate` to the `OptionalFields` list in a `PutBucketInventoryConfiguration` request. For more information, see Cataloging and analyzing your data with S3 Inventory.
- **ETag** – The entity tag (ETag) is a hash of the object. The ETag reflects changes only to the contents of an object, not to its metadata. The ETag might or might not be an MD5 digest of the object data. Whether it is depends on how the object was created and how it

is encrypted. For more information, see Object in the *Amazon Simple Storage Service API Reference*.

- **Checksum algorithm** – Indicates the algorithm that is used to create the checksum for the object. Supported values include CRC64NVME, CRC32, CRC32C, SHA1, SHA256, MD5, XXHASH64, XXHASH3, XXHASH128, and SHA512. For more information, see Using supported checksum algorithms.
- **All Object Lock configurations** – The Object Lock status of the object, including the following settings:
  - **Object Lock: Retention mode** – The level of protection applied to the object, either *Governance* or *Compliance*.
  - **Object Lock: Retain until date** – The date until which the locked object cannot be deleted.
  - **Object Lock: Legal hold status** – The legal hold status of the locked object.
  - **Object Lock: Event hold status** – The event hold status of the object.
  - **Object Lock: Event hold duration** – The variable retention duration that is configured for the object. Selecting this field also adds an **Object Lock: Event hold duration unit** column (DAYS or YEARS) to the report.

  For information about S3 Object Lock, see How S3 Object Lock works.
- **Lifecycle Expiration Date** – The lifecycle expiration timestamp for objects in your Inventory report. This field will only be populated, if the object is to be expired by an applicable lifecycle rule. In other cases, the field will be empty. For more information, see Expiring objects.

For more information about the contents of an inventory report, see Amazon S3 Inventory list.

For more information about restricting access to certain optional metadata fields in an inventory configuration, see Control S3 Inventory report configuration creation.

15. Choose **Create**.

Using the REST API to work with S3 Inventory

The following are the REST operations that you can use to work with Amazon S3 Inventory.

- DeleteBucketInventoryConfiguration

- GetBucketInventoryConfiguration
- ListBucketInventoryConfigurations
- PutBucketInventoryConfiguration

Locating your inventory list

When an inventory list is published, the manifest files are published to the following location in the destination bucket.

```
destination-prefix/amzn-s3-demo-source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/manifest.json
 destination-prefix/amzn-s3-demo-source-bucket/config-ID/YYYY-MM-DDTHH-MMZ/
manifest.checksum
 destination-prefix/amzn-s3-demo-source-bucket/config-ID/hive/dt=YYYY-MM-DD-HH-MM/
symlink.txt
```

- *destination-prefix* is the object key name prefix that is optionally specified in the inventory configuration. You can use this prefix to group all the inventory list files in a common location within the destination bucket.
- *amzn-s3-demo-source-bucket* is the source bucket that the inventory list is for. The source bucket name is added to prevent collisions when multiple inventory reports from different source buckets are sent to the same destination bucket.
- *config-ID* is added to prevent collisions with multiple inventory reports from the same source bucket that are sent to the same destination bucket. The *config-ID* comes from the inventory report configuration, and is the name for the report that is defined during setup.
- *YYYY-MM-DDTHH-MMZ* is the timestamp that consists of the start time and the date when the inventory report generation process begins scanning the bucket; for example, `2016-11-06T21-32Z`.
- `manifest.json` is the manifest file.
- `manifest.checksum` is the MD5 hash of the content of the `manifest.json` file.
- `symlink.txt` is the Apache Hive-compatible manifest file.

The inventory lists are published daily or weekly to the following location in the destination bucket.

```
destination-prefix/amzn-s3-demo-source-bucket/config-ID/data/example-file-name.csv.gz
...
```

```
destination-prefix/amzn-s3-demo-source-bucket/config-ID/data/example-file-name-1.csv.gz
```

- *destination-prefix* is the object key name prefix that is optionally specified in the inventory configuration. You can use this prefix to group all the inventory list files in a common location in the destination bucket.
- *amzn-s3-demo-source-bucket* is the source bucket that the inventory list is for. The source bucket name is added to prevent collisions when multiple inventory reports from different source buckets are sent to the same destination bucket.
- *example-file-name*`.csv.gz` is one of the CSV inventory files. ORC inventory names end with the file name extension `.orc`, and Parquet inventory names end with the file name extension `.parquet`.

Inventory manifest

The manifest files `manifest.json` and `symlink.txt` describe where the inventory files are located. Whenever a new inventory list is delivered, it is accompanied by a new set of manifest files. These files might overwrite each other. In versioning-enabled buckets, Amazon S3 creates new versions of the manifest files.

Each manifest contained in the `manifest.json` file provides metadata and other basic information about an inventory. This information includes the following:

- The source bucket name
- The destination bucket name
- The version of the inventory
- The creation timestamp in the epoch date format that consists of the start time and the date when the inventory report generation process begins scanning the bucket
- The format and schema of the inventory files
- A list of the inventory files that are in the destination bucket

Whenever a `manifest.json` file is written, it is accompanied by a `manifest.checksum` file that is the MD5 hash of the content of the `manifest.json` file.

Example Inventory manifest in a `manifest.json` file

The following examples show an inventory manifest in a `manifest.json` file for CSV, ORC, and Parquet-formatted inventories.

CSV

The following is an example of a manifest in a `manifest.json` file for a CSV-formatted inventory.

```
{
    "sourceBucket": "amzn-s3-demo-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-inventory-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "CSV",
    "fileSchema": "Bucket, Key, VersionId, IsLatest, IsDeleteMarker,
 Size, LastModifiedDate, ETag, StorageClass, IsMultipartUploaded,
 ReplicationStatus, EncryptionStatus, ObjectLockRetainUntilDate,
 ObjectLockMode, ObjectLockLegalHoldStatus, IntelligentTieringAccessTier,
 BucketKeyStatus, ChecksumAlgorithm, ObjectAccessControlList, ObjectOwner,
 IntelligentTieringReferenceDate",
    "files": [
        {
            "key": "Inventory/amzn-s3-demo-source-bucket/2016-11-06T21-32Z/
files/939c6d46-85a9-4ba8-87bd-9db705a579ce.csv.gz",
            "size": 2147483647,
            "MD5checksum": "f11166069f1990abeb9c97ace9cdfabc"
        }
    ]
}
```

ORC

The following is an example of a manifest in a `manifest.json` file for an ORC-formatted inventory.

```
{
    "sourceBucket": "amzn-s3-demo-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "ORC",
```

```
    "fileSchema":
 "struct<bucket:string,key:string,version_id:string,is_latest:boolean,is_delete_marker:boole
    "files": [
        {
            "key": "inventory/amzn-s3-demo-source-bucket/data/
d794c570-95bb-4271-9128-26023c8b4900.orc",
            "size": 56291,
            "MD5checksum": "5925f4e78e1695c2d020b9f6eexample"
        }
    ]
}
```

Parquet

The following is an example of a manifest in a `manifest.json` file for a Parquet-formatted inventory.

```
{
    "sourceBucket": "amzn-s3-demo-source-bucket",
    "destinationBucket": "arn:aws:s3:::example-destination-bucket",
    "version": "2016-11-30",
    "creationTimestamp" : "1514944800000",
    "fileFormat": "Parquet",
    "fileSchema": "message s3.inventory { required binary bucket (UTF8);
 required binary key (UTF8); optional binary version_id (UTF8); optional boolean
 is_latest; optional boolean is_delete_marker; optional int64 size; optional
 int64 last_modified_date (TIMESTAMP_MILLIS); optional binary e_tag (UTF8);
 optional binary storage_class (UTF8); optional boolean is_multipart_uploaded;
 optional binary replication_status (UTF8); optional binary encryption_status
 (UTF8); optional int64 object_lock_retain_until_date (TIMESTAMP_MILLIS); optional
 binary object_lock_mode (UTF8); optional binary object_lock_legal_hold_status
 (UTF8); optional binary intelligent_tiering_access_tier (UTF8); optional binary
 bucket_key_status (UTF8); optional binary checksum_algorithm (UTF8); optional
 binary object_access_control_list (UTF8); optional binary object_owner (UTF8);
 optional int64 intelligent_tiering_reference_date (TIMESTAMP_MILLIS);}",
    "files": [
        {
           "key": "inventory/amzn-s3-demo-source-bucket/data/
d754c470-85bb-4255-9218-47023c8b4910.parquet",
            "size": 56291,
            "MD5checksum": "5825f2e18e1695c2d030b9f6eexample"
        }
    ]
```

```
}
```

The `symlink.txt` file is an Apache Hive-compatible manifest file that allows Hive to automatically discover inventory files and their associated data files. The Hive-compatible manifest works with the Hive-compatible services Athena and Amazon Redshift Spectrum. It also works with Hive-compatible applications, including Presto, Apache Hive, Apache Spark, and many others.

> **Important**
>
> The `symlink.txt` Apache Hive-compatible manifest file does not currently work with AWS Glue.
>
> Reading the `symlink.txt` file with Apache Hive and Apache Spark is not supported for ORC and Parquet-formatted inventory files.

Setting up Amazon S3 Event Notifications for inventory completion

You can set up an Amazon S3 event notification to receive notice when the manifest checksum file is created, which indicates that an inventory list has been added to the destination bucket. The manifest is an up-to-date list of all the inventory lists at the destination location.

Amazon S3 can publish events to an Amazon Simple Notification Service (Amazon SNS) topic, an Amazon Simple Queue Service (Amazon SQS) queue, or an AWS Lambda function. For more information, see Amazon S3 Event Notifications.

The following notification configuration defines that all `manifest.checksum` files newly added to the destination bucket are processed by the AWS Lambda `cloud-function-list-write`.

```
<NotificationConfiguration>
  <QueueConfiguration>
      <Id>1</Id>
      <Filter>
          <S3Key>
              <FilterRule>
                  <Name>prefix</Name>
                  <Value>destination-prefix/source-bucket</Value>
              </FilterRule>
              <FilterRule>
                  <Name>suffix</Name>
                  <Value>checksum</Value>
```

```
                </FilterRule>
            </S3Key>
        </Filter>
        <CloudFunction>arn:aws:lambda:us-west-2:222233334444:cloud-function-list-write</
CloudFunction>
        <Event>s3:ObjectCreated:*</Event>
    </QueueConfiguration>
    </NotificationConfiguration>
```

For more information, see Using AWS Lambda with Amazon S3 in the *AWS Lambda Developer Guide*.

Querying Amazon S3 Inventory with Amazon Athena

You can query Amazon S3 Inventory files with standard SQL queries by using Amazon Athena in all Regions where Athena is available. To check for AWS Region availability, see the AWS Region Table.

Athena can query Amazon S3 Inventory files in Apache optimized row columnar (ORC), Apache Parquet, or comma-separated values (CSV) format. For best query speed and lower costs, use ORC-formatted or Parquet-formatted inventory files. ORC and Parquet are columnar file formats designed for Apache Hadoop. The columnar format lets the reader read, decompress, and process only the columns that are required for the current query. The ORC and Parquet formats for Amazon S3 Inventory are available in all AWS Regions.

To use Athena to query Amazon S3 Inventory files

1. Create an Athena table. For information about creating a table, see Creating Tables in Amazon Athena in the *Amazon Athena User Guide*.
2. Create your query by using one of the following sample query templates, depending on whether you're querying an ORC-formatted, a Parquet-formatted, or a CSV-formatted inventory report.

   - When you're using Athena to query an ORC-formatted inventory report, use the following sample query as a template.

     The following sample query includes all the optional fields in an ORC-formatted inventory report.

     To use this sample query, do the following:

     - Replace *your_table_name* with the name of the Athena table that you created.

- Remove any optional fields that you did not choose for your inventory so that the query corresponds to the fields chosen for your inventory.
- Replace the following bucket name and inventory location (the configuration ID) as appropriate for your configuration.

  `s3://amzn-s3-demo-bucket/config-ID/hive/`
- Replace the *2022-01-01-00-00* date under `projection.dt.range` with the first day of the time range within which you partition the data in Athena. For more information, see Partitioning data in Athena.

```
CREATE EXTERNAL TABLE your_table_name (
         bucket string,
         key string,
         version_id string,
         is_latest boolean,
         is_delete_marker boolean,
         size bigint,
         last_modified_date timestamp,
         e_tag string,
         storage_class string,
         is_multipart_uploaded boolean,
         replication_status string,
         encryption_status string,
         object_lock_retain_until_date bigint,
         object_lock_mode string,
         object_lock_legal_hold_status string,
         object_lock_event_hold_status string,
         object_lock_event_hold_duration int,
         object_lock_event_hold_duration_unit string,
         intelligent_tiering_access_tier string,
         bucket_key_status string,
         checksum_algorithm string,
         object_access_control_list string,
         object_owner string,
         lifecycle_expiration_date timestamp
) PARTITIONED BY (
        dt string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.orc.OrcSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
```

```
  LOCATION 's3://amzn-s3-demo-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

- When you're using Athena to query a Parquet-formatted inventory report, use the sample query for an ORC-formatted report. However, use the following Parquet SerDe in place of the ORC SerDe in the `ROW FORMAT SERDE` statement.

  ```
  ROW FORMAT SERDE 'org.apache.hadoop.hive.ql.io.parquet.serde.ParquetHiveSerDe'
  ```

- When you're using Athena to query a CSV-formatted inventory report, use the following sample query as a template.

  The following sample query includes all the optional fields in an CSV-formatted inventory report.

  To use this sample query, do the following:

  - Replace *`your_table_name`* with the name of the Athena table that you created.
  - Remove any optional fields that you did not choose for your inventory so that the query corresponds to the fields chosen for your inventory.
  - Replace the following bucket name and inventory location (the configuration ID) as appropriate for your configuration.

    `s3://amzn-s3-demo-bucket/config-ID/hive/`

  - Replace the *`2022-01-01-00-00`* date under `projection.dt.range` with the first day of the time range within which you partition the data in Athena. For more information, see Partitioning data in Athena.

  ```
  CREATE EXTERNAL TABLE your_table_name (
           bucket string,
           key string,
           version_id string,
           is_latest boolean,
           is_delete_marker boolean,
  ```

```
         size string,
         last_modified_date string,
         e_tag string,
         storage_class string,
         is_multipart_uploaded boolean,
         replication_status string,
         encryption_status string,
         object_lock_retain_until_date string,
         object_lock_mode string,
         object_lock_legal_hold_status string,
         object_lock_event_hold_status string,
         object_lock_event_hold_duration string,
         object_lock_event_hold_duration_unit string,
         intelligent_tiering_access_tier string,
         bucket_key_status string,
         checksum_algorithm string,
         object_access_control_list string,
         object_owner string,
         lifecycle_expiration_date string
) PARTITIONED BY (
        dt string
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
  STORED AS INPUTFORMAT 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat'
  OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat'
  LOCATION 's3://amzn-s3-demo-bucket/config-ID/hive/'
  TBLPROPERTIES (
    "projection.enabled" = "true",
    "projection.dt.type" = "date",
    "projection.dt.format" = "yyyy-MM-dd-HH-mm",
    "projection.dt.range" = "2022-01-01-00-00,NOW",
    "projection.dt.interval" = "1",
    "projection.dt.interval.unit" = "HOURS"
  );
```

3. You can now run various queries on your inventory, as shown in the following examples. Replace each *user input placeholder* with your own information.

```
# Get a list of the latest inventory report dates available.
SELECT DISTINCT dt FROM your_table_name ORDER BY 1 DESC limit 10;

# Get the encryption status for a provided report date.
```

```
SELECT encryption_status, count(*) FROM your_table_name WHERE dt = 'YYYY-MM-DD-HH-
MM' GROUP BY encryption_status;

# Get the encryption status for inventory report dates in the provided range.
SELECT dt, encryption_status, count(*) FROM your_table_name
WHERE dt > 'YYYY-MM-DD-HH-MM' AND dt < 'YYYY-MM-DD-HH-MM' GROUP BY dt,
 encryption_status;
```

When you configure S3 Inventory to add the Object Access Control List (Object ACL) field to an inventory report, the report displays the value for the Object ACL field as a base64-encoded string. To get the decoded value in JSON for the Object ACL field, you can query this field by using Athena. See the following query examples. For more information about the Object ACL field, see Working with the Object ACL field.

```
# Get the S3 keys that have Object ACL grants with public access.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'uri') = 'http://acs.amazonaws.com/groups/global/AllUsers'
```

```
# Get the S3 keys that have Object ACL grantees in addition to the object owner.
WITH grants AS
    (SELECT key,
    from_utf8(from_base64(object_access_control_list)) AS
 object_access_control_list,
         object_owner,
         CAST(json_extract(from_utf8(from_base64(object_access_control_list)),
         '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))) AS grants_array
    FROM your_table_name)
SELECT key,
       grant,
       objectowner
```

```
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE cardinality(grants_array) > 1 AND element_at(grant, 'canonicalId') !=
 object_owner;
```

```
# Get the S3 keys with READ permission that is granted in the Object ACL.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'permission') = 'READ';
```

```
# Get the S3 keys that have Object ACL grants to a specific canonical user ID.
WITH grants AS (
    SELECT key,
        CAST(
            json_extract(from_utf8(from_base64(object_access_control_list)),
 '$.grants') AS ARRAY(MAP(VARCHAR, VARCHAR))
        ) AS grants_array
    FROM your_table_name
)
SELECT key,
       grants_array,
       grant
FROM grants, UNNEST(grants_array) AS t(grant)
WHERE element_at(grant, 'canonicalId') = 'user-canonical-id';
```

```
# Get the number of grantees on the Object ACL.
SELECT key,
       object_access_control_list,
       json_array_length(json_extract(object_access_control_list,'$.grants')) AS
 grants_count
```

```
FROM your_table_name;
```

For more information about using Athena, see the Amazon Athena User Guide.

Converting empty version ID strings in Amazon S3 Inventory reports to null strings

> **Note**
>
> **The following procedure applies only to Amazon S3 Inventory reports that include all versions, and only if the "all versions" reports are used as manifests for S3 Batch Operations on buckets that have S3 Versioning enabled.** You are not required to convert strings for S3 Inventory reports that specify the current version only.

You can use S3 Inventory reports as manifests for S3 Batch Operations. However, when S3 Versioning is enabled on a bucket, S3 Inventory reports that include all versions mark any null-versioned objects with empty strings in the version ID field. When an Inventory Report includes all object version IDs, Batch Operations recognizes `null` strings as version IDs, but not empty strings.

When an S3 Batch Operations job uses an "all versions" S3 Inventory report as a manifest, it fails all tasks on objects that have an empty string in the version ID field. To convert empty strings in the version ID field of the S3 Inventory report to `null` strings for Batch Operations, use the following procedure.

Update an Amazon S3 Inventory report for use with Batch Operations

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Navigate to your S3 Inventory report. The inventory report is located in the destination bucket that you specified while configuring your inventory report. For more information about locating inventory reports, see Locating your inventory list.
   a. Choose the destination bucket.
   b. Choose the folder. The folder is named after the original source bucket.
   c. Choose the folder named after the inventory configuration.

d. Select the check box next to the folder named **hive**. Choose **Copy S3 URI** to copy the S3 URI for the folder.

3. Open the Amazon Athena console at https://console.aws.amazon.com/athena/.
4. In the query editor, choose **Settings**, then choose **Manage**. On the **Manage settings** page, for **Location of query result**, choose an S3 bucket to store your query results in.
5. In the query editor, create an Athena table to hold the data in the inventory report using the following command. Replace *table_name* with a name of your choosing, and in the LOCATION clause, insert the S3 URI that you copied earlier. Then choose **Run** to run the query.

```
CREATE EXTERNAL TABLE table_name(bucket string, key string,
 version_id string) PARTITIONED BY (dt string)ROW FORMAT SERDE
 'org.apache.hadoop.hive.serde2.OpenCSVSerde'STORED AS INPUTFORMAT
 'org.apache.hadoop.hive.ql.io.SymlinkTextInputFormat' OUTPUTFORMAT
 'org.apache.hadoop.hive.ql.io.IgnoreKeyTextOutputFormat' LOCATION 'Copied S3 URI';
```

6. To clear the query editor, choose **Clear**. Then load the inventory report into the table using the following command. Replace *table_name* with the one that you chose in the prior step. Then choose **Run** to run the query.

```
MSCK REPAIR TABLE table_name;
```

7. To clear the query editor, choose **Clear**. Run the following SELECT query to retrieve all entries in the original inventory report and replace any empty version IDs with null strings. Replace *table_name* with the one that you chose earlier, and replace *YYYY-MM-DD-HH-MM* in the WHERE clause with the date of the inventory report that you want this tool to run on. Then choose **Run** to run the query.

```
SELECT bucket as Bucket, key as Key, CASE WHEN version_id = '' THEN 'null' ELSE
 version_id END as VersionId FROM table_name WHERE dt = 'YYYY-MM-DD-HH-MM';
```

8. Return to the Amazon S3 console (https://console.aws.amazon.com/s3/), and navigate to the S3 bucket that you chose for **Location of query result** earlier. Inside, there should be a series of folders ending with the date.

   For example, you should see something like **s3://*amzn-s3-demo-bucket*/*query-result-location*/Unsaved/2021/10/07/**. You should see .csv files containing the results of the SELECT query that you ran.

Choose the CSV file with the latest modified date. Download this file to your local machine for the next step.

9. The generated CSV file contains a header row. To use this CSV file as input for an S3 Batch Operations job, you must remove the header row, because Batch Operations doesn't support header rows on CSV manifests.

   To remove the header row, you can run one of the following commands on the file. Replace *`file.csv`* with the name of your CSV file.

   For macOS and Linux machines, run the `tail` command in a Terminal window.

   ```
   tail -n +2 file.csv > tmp.csv && mv tmp.csv file.csv
   ```

   For Windows machines, run the following script in a Windows PowerShell window. Replace *`File-location`* with the path to your file, and *`file.csv`* with the file name.

   ```
   $ins = New-Object System.IO.StreamReader File-location\file.csv
   $outs = New-Object System.IO.StreamWriter File-location\temp.csv
   try {
       $skip = 0
       while ( !$ins.EndOfStream ) {
           $line = $ins.ReadLine();
           if ( $skip -ne 0 ) {
               $outs.WriteLine($line);
           } else {
               $skip = 1
           }
       }
   } finally {
       $outs.Close();
       $ins.Close();
   }
   Move-Item File-location\temp.csv File-location\file.csv -Force
   ```

10. After removing the header row from the CSV file, you are ready to use it as a manifest in an S3 Batch Operations job. Upload the CSV file to an S3 bucket or location of your choosing, and then create a Batch Operations job using the CSV file as the manifest.

    For more information about creating a Batch Operations job, see Creating an S3 Batch Operations job.

Working with the Object ACL field

An Amazon S3 Inventory report contains a list of the objects in the S3 source bucket and metadata for each object. The Object access control list (ACL) field is a metadata field that is available in Amazon S3 Inventory. Specifically, the Object ACL field contains the access control list (ACL) for each object. The ACL for an object defines which AWS accounts or groups are granted access to this object and the type of access that is granted. For more information, see Access control list (ACL) overview and Amazon S3 Inventory list.

The Object ACL field in Amazon S3 Inventory reports is defined in JSON format. The JSON data includes the following fields:

- `version` – The version of the Object ACL field format in the inventory reports. It's in date format `yyyy-mm-dd`.
- `status` – Possible values are `AVAILABLE` or `UNAVAILABLE` to indicate whether an Object ACL is available for an object. When the status for the Object ACL is `UNAVAILABLE`, the value of the Object Owner field in the inventory report is also `UNAVAILABLE`.
- `grants` – Grantee-permission pairs that list the permission status of each grantee that is granted by the Object ACL. The available values for a grantee are `CanonicalUser` and `Group`. For more information about grantees, see Grantees in access control lists.

  For a grantee with the `Group` type, a grantee-permission pair includes the following attributes:

  - `uri` – A predefined Amazon S3 group.
  - `permission` – The ACL permissions that are granted on the object. For more information, see ACL permissions on an object.
  - `type` – The type `Group`, which denotes that the grantee is group.

  For a grantee with the `CanonicalUser` type, a grantee-permission pair includes the following attributes:

  - `canonicalId` – An obfuscated form of the AWS account ID. The canonical user ID for an AWS account is specific to that account. You can retrieve the canonical user ID. For more information, see Find the canonical user ID for your AWS account in the *AWS Account Management Reference Guide*.
  - `permission` – The ACL permissions that are granted on the object. For more information, see ACL permissions on an object.
  - `type` – The type `CanonicalUser`, which denotes that the grantee is an AWS account.

The following example shows possible values for the Object ACL field in JSON format:

```
{
    "version": "2022-11-10",
    "status": "AVAILABLE",
    "grants": [{
        "uri": "http://acs.amazonaws.com/groups/global/AllUsers",
        "permission": "READ",
        "type": "Group"
    }, {
        "canonicalId": "example-canonical-id",
        "permission": "FULL_CONTROL",
        "type": "CanonicalUser"
    }]
}
```

Note

The Object ACL field is defined in JSON format. An inventory report displays the value for the Object ACL field as a base64-encoded string.
For example, suppose that you have the following Object ACL field in JSON format:

```
{
        "version": "2022-11-10",
        "status": "AVAILABLE",
        "grants": [{
            "canonicalId": "example-canonical-user-ID",
            "type": "CanonicalUser",
            "permission": "READ"
        }]
}
```

The Object ACL field is encoded and shown as the following base64-encoded string:

```
eyJ2ZXJzaW9uIjoiMjAyMi0xMS0xMCIsInN0YXR1cyI6IkFWQUlMQUJMRSIsImdyYW50cyI6W3siY2Fub25pY2FsSW
```

To get the decoded value in JSON for the Object ACL field, you can query this field in Amazon Athena. For query examples, see Querying Amazon S3 Inventory with Amazon Athena.

Best practices design patterns: optimizing Amazon S3 performance

Your applications can easily achieve thousands of transactions per second in request performance when uploading and retrieving storage from Amazon S3. Amazon S3 automatically scales to high request rates. For example, your application can achieve at least 3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per partitioned Amazon S3 prefix. There are no limits to the number of prefixes in a bucket. You can increase your read or write performance by using parallelization. For example, if you create 10 prefixes in an Amazon S3 bucket to parallelize reads, you could scale your read performance to 55,000 read requests per second. Similarly, you can scale write operations by writing to multiple prefixes. The scaling, in the case of both read and write operations, happens gradually and is not instantaneous, and actual performance will vary based on your specific workload characteristics, usage patterns, and system configuration. While Amazon S3 is scaling to your new higher request rate, you may see some 503 (Slow Down) errors. These errors will dissipate when the scaling is complete. For more information about creating and using prefixes, see Organizing objects using prefixes.

Some data lake applications on Amazon S3 scan millions or billions of objects for queries that run over petabytes of data. These data lake applications achieve single-instance transfer rates that maximize the network interface use for their Amazon EC2 instance, which can be up to 100 Gb/s on a single instance. These applications then aggregate throughput across multiple instances to get multiple terabits per second.

Other applications are sensitive to latency, such as social media messaging applications. These applications can achieve consistent small object latencies (and first-byte-out latencies for larger objects) of roughly 100–200 milliseconds.

Other AWS services can also help accelerate performance for different application architectures. For example, if you want higher transfer rates over a single HTTP connection or single-digit millisecond latencies, use Amazon CloudFront or Amazon ElastiCache for caching with Amazon S3.

Additionally, if you want fast data transport over long distances between a client and an S3 bucket, use Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration. Transfer Acceleration uses the globally distributed edge locations in CloudFront to accelerate data transport over geographical distances. If your Amazon S3 workload uses server-side encryption with AWS KMS, see AWS KMS Limits in the AWS Key Management Service Developer Guide for information about the request rates supported for your use case.

The following topics describe best practice guidelines and design patterns for optimizing performance for applications that use Amazon S3. Refer to the Performance guidelines for Amazon S3 and Performance design patterns for Amazon S3 for the most current information about performance optimization for Amazon S3.

> **Note**
>
> For more information about using the Amazon S3 Express One Zone storage class with directory buckets, see S3 Express One Zone and Working with directory buckets.

Topics

- Performance guidelines for Amazon S3
- Performance design patterns for Amazon S3

Performance guidelines for Amazon S3

When building applications that upload and retrieve objects from Amazon S3, follow our best practices guidelines to optimize performance. We also offer more detailed Performance design patterns for Amazon S3 .

To obtain the best performance for your application on Amazon S3, we recommend the following guidelines.

Topics

- Measure performance
- Scale storage connections horizontally
- Use byte-range fetches
- Retry requests for latency-sensitive applications
- Combine Amazon S3 (Storage) and Amazon EC2 (compute) in the same AWS Region
- Use Amazon S3 Transfer Acceleration to minimize latency caused by distance
- Use the latest version of the AWS SDKs

Measure performance

When optimizing performance, look at network throughput, CPU, and DRAM requirements. Depending on the mix of demands for these different resources, it might be worth evaluating different Amazon EC2 instance types. For more information about instance types, see Instance Types in the *Amazon EC2 User Guide*.

It's also helpful to look at DNS lookup time, latency, and data transfer speed using HTTP analysis tools when measuring performance.

To understand the performance requirements and optimize the performance of your application, you can also monitor the 503 error responses that you receive. Monitoring certain performance metrics may incur additional expenses. For more information, see Amazon S3 pricing.

Monitor the number of 503 (Slow Down) status error responses

To monitor the number of 503 status error responses that you get, you can use one of the following options:

- Use Amazon CloudWatch request metrics for Amazon S3. The CloudWatch request metrics include a metric for 5xx status responses. For more information about CloudWatch request metrics, see Monitoring metrics with Amazon CloudWatch.
- Use the 503 (Service Unavailable) error count available in the advanced metrics section of Amazon S3 Storage Lens. For more information, see Using S3 Storage Lens metrics to improve performance.
- Use Amazon S3 server access logging. With server access logging, you can filter and review all requests that receive 503 (Internal Error) responses. You can also use Amazon Athena to parse logs. For more information about server access logging, see Logging requests with server access logging.

By monitoring the number of HTTP 503 status error code, you can often gain valuable insights into which prefixes, keys, or buckets are getting the most throttling requests.

Scale storage connections horizontally

Spreading requests across many connections is a common design pattern to horizontally scale performance. When you build high performance applications, think of Amazon S3 as a very large distributed system, not as a single network endpoint like a traditional storage server. You can achieve the best performance by issuing multiple concurrent requests to Amazon S3. Spread

these requests over separate connections to maximize the accessible bandwidth from Amazon S3. Amazon S3 doesn't have any limits for the number of connections made to your bucket.

Use byte-range fetches

Using the Range HTTP header in a GET Object request, you can fetch a byte-range from an object, transferring only the specified portion. You can use concurrent connections to Amazon S3 to fetch different byte ranges from within the same object. This helps you achieve higher aggregate throughput versus a single whole-object request. Fetching smaller ranges of a large object also allows your application to improve retry times when requests are interrupted. For more information, see Downloading objects.

If objects are PUT using a multipart upload, it's a good practice to GET them in the same part sizes (or at least aligned to part boundaries) for best performance. GET requests can directly address individual parts; for example, `GET ?partNumber=N`.

Retry requests for latency-sensitive applications

Aggressive timeouts and retries help drive consistent latency. Given the large scale of Amazon S3, if the first request is slow, a retried request is likely to take a different path and quickly succeed. The AWS SDKs have configurable timeout and retry values that you can tune to the tolerances of your specific application.

Combine Amazon S3 (Storage) and Amazon EC2 (compute) in the same AWS Region

Although S3 bucket names are globally unique, each bucket is stored in a Region that you select when you create the bucket. To learn more about bucket naming guidelines, see Buckets overview and Bucket naming rules. To optimize performance, we recommend that you access the bucket from Amazon EC2 instances in the same AWS Region when possible. This helps reduce network latency and data transfer costs.

For more information about data transfer costs, see Amazon S3 Pricing.

Use Amazon S3 Transfer Acceleration to minimize latency caused by distance

Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration manages fast, easy, and secure transfers of files over long geographic distances between the client and an S3 bucket.

Transfer Acceleration takes advantage of the globally distributed edge locations in Amazon CloudFront. As the data arrives at an edge location, it is routed to Amazon S3 over an optimized network path. Transfer Acceleration is ideal for transferring gigabytes to terabytes of data regularly across continents. It's also useful for clients that upload to a centralized bucket from all over the world.

You can use the Amazon S3 Transfer Acceleration Speed comparison tool to compare accelerated and non-accelerated upload speeds across Amazon S3 Regions. The Speed Comparison tool uses multipart uploads to transfer a file from your browser to various Amazon S3 Regions with and without using Amazon S3 Transfer Acceleration.

Use the latest version of the AWS SDKs

The AWS SDKs provide built-in support for many of the recommended guidelines for optimizing Amazon S3 performance. The SDKs provide a simpler API for taking advantage of Amazon S3 from within an application and are regularly updated to follow the latest best practices. For example, the SDKs include logic to automatically retry requests on HTTP 503 errors and are investing in code to respond and adapt to slow connections.

The SDKs also provide the Transfer Manager, which automates horizontally scaling connections to achieve thousands of requests per second, using byte-range requests where appropriate. It's important to use the latest version of the AWS SDKs to obtain the latest performance optimization features.

You can also optimize performance when you are using HTTP REST API requests. When using the REST API, you should follow the same best practices that are part of the SDKs. Allow for timeouts and retries on slow requests, and multiple connections to allow fetching of object data in parallel. For information about using the REST API, see the Amazon Simple Storage Service API Reference.

Performance design patterns for Amazon S3

When designing applications to upload and retrieve objects from Amazon S3, use our best practices design patterns for achieving the best performance for your application. We also offer Performance guidelines for Amazon S3 for you to consider when planning your application architecture.

To optimize performance, you can use the following design patterns.

Topics

- Using caching for frequently accessed content
- Timeouts and retries for latency-sensitive applications
- Horizontal scaling and request parallelization for high throughput
- Using Amazon S3 Transfer Acceleration to accelerate geographically disparate data transfers
- Optimizing for high-request rate workloads

Using caching for frequently accessed content

Many applications that store data in Amazon S3 serve a "working set" of data that is repeatedly requested by users. If a workload is sending repeated GET requests for a common set of objects, you can use a cache such as Amazon CloudFront or Amazon ElastiCache to optimize performance. Successful cache adoption can result in low latency and high data transfer rates. Applications that use caching also send fewer direct requests to Amazon S3, which can help reduce request costs.

Amazon CloudFront is a fast content delivery network (CDN) that transparently caches data from Amazon S3 in a large set of geographically distributed points of presence (PoPs). When objects might be accessed from multiple Regions, or over the internet, CloudFront allows data to be cached close to the users that are accessing the objects. This can result in high performance delivery of popular Amazon S3 content. For information about CloudFront, see the Amazon CloudFront Developer Guide.

Amazon ElastiCache is a managed, in-memory cache. With ElastiCache, you can provision Amazon EC2 instances that cache objects in memory. This caching results in orders of magnitude reduction in GET latency and substantial increases in download throughput. To use ElastiCache, you modify application logic to both populate the cache with hot objects and check the cache for hot objects before requesting them from Amazon S3. For examples of using ElastiCache to improve Amazon S3 GET performance, see the blog post Turbocharge Amazon S3 with Amazon ElastiCache for Redis.

Timeouts and retries for latency-sensitive applications

There are certain situations where an application receives a response from Amazon S3 indicating that a retry is necessary. Amazon S3 maps bucket and object names to the object data associated with them. If an application generates high request rates (typically sustained rates of over 5,000 requests per second to a small number of objects), it might receive HTTP 503 *slowdown* responses. If these errors occur, each AWS SDK implements automatic retry logic using exponential backoff. If you are not using an AWS SDK, you should implement retry logic when receiving the HTTP 503

error. For information about back-off techniques, see Retry behavior in the *AWS SDKs and Tools Reference Guide*.

Amazon S3 automatically scales in response to sustained new request rates, dynamically optimizing performance. While Amazon S3 is internally optimizing for a new request rate, you will receive HTTP 503 request responses temporarily until the optimization completes. After Amazon S3 internally optimizes performance for the new request rate, all requests are generally served without retries.

For latency-sensitive applications, Amazon S3 advises tracking and aggressively retrying slower operations. When you retry a request, we recommend using a new connection to Amazon S3 and performing a fresh DNS lookup.

When you make large variably sized requests (for example, more than 128 MB), we advise tracking the throughput being achieved and retrying the slowest 5 percent of the requests. When you make smaller requests (for example, less than 512 KB), where median latencies are often in the tens of milliseconds range, a good guideline is to retry a GET or PUT operation after 2 seconds. If additional retries are needed, the best practice is to back off. For example, we recommend issuing one retry after 2 seconds and a second retry after an additional 4 seconds.

If your application makes fixed-size requests to Amazon S3, you should expect more consistent response times for each of these requests. In this case, a simple strategy is to identify the slowest 1 percent of requests and to retry them. Even a single retry is frequently effective at reducing latency.

If you are using AWS Key Management Service (AWS KMS) for server-side encryption, see Quotas in the *AWS Key Management Service Developer Guide* for information about the request rates that are supported for your use case.

Horizontal scaling and request parallelization for high throughput

Amazon S3 is a very large distributed system. To help you take advantage of its scale, we encourage you to horizontally scale parallel requests to the Amazon S3 service endpoints. In addition to distributing the requests within Amazon S3, this type of scaling approach helps distribute the load over multiple paths through the network.

For high-throughput transfers, Amazon S3 advises using applications that use multiple connections to GET or PUT data in parallel. For example, this is supported by Amazon S3 Transfer Manager in the AWS Java SDK, and most of the other AWS SDKs provide similar constructs. For some

applications, you can achieve parallel connections by launching multiple requests concurrently in different application threads, or in different application instances. The best approach to take depends on your application and the structure of the objects that you are accessing.

You can use the AWS SDKs to issue GET and PUT requests directly rather than employing the management of transfers in the AWS SDK. This approach lets you tune your workload more directly, while still benefiting from the SDK's support for retries and its handling of any HTTP 503 responses that might occur. As a general rule, when you download large objects from Amazon S3, we suggest making concurrent requests to maximize network throughput and optimize download performance. You can achieve this by either requesting specific byte ranges of the object or downloading individual parts of a multipart object simultaneously. This parallel download approach helps fully use your network interface card (NIC) capacity. For objects that were uploaded using multipart upload, we recommend downloading them using the same part sizes or aligning requests to the original part boundaries for best performance. This method of concurrent downloading provides higher aggregate throughput compared to single whole-object requests.

Measuring performance is important when you tune the number of requests to issue concurrently. We recommend starting with a single request at a time. Measure the network bandwidth being achieved and the use of other resources that your application uses in processing the data. You can then identify the bottleneck resource (that is, the resource with the highest usage), and hence the number of requests that are likely to be useful. For example, if processing one request at a time leads to a CPU usage of 25 percent, it suggests that up to four concurrent requests can be accommodated. Measurement is essential, and it is worth confirming resource use as the request rate is increased.

If your application issues requests directly to Amazon S3 using the REST API, we recommend using a pool of HTTP connections and re-using each connection for a series of requests. Avoiding per-request connection setup removes the need to perform TCP slow-start and Secure Sockets Layer (SSL) handshakes on each request. For information about using the REST API, see the Amazon Simple Storage Service API Reference.

Finally, it's worth paying attention to DNS and double-checking that requests are being spread over a wide pool of Amazon S3 IP addresses. DNS queries for Amazon S3 cycle through a large list of IP endpoints. But caching resolvers or application code that reuses a single IP address do not benefit from address diversity and the load balancing that follows from it. Network utility tools such as the `netstat` command line tool can show the IP addresses being used for communication with Amazon S3, and we provide guidelines for DNS configurations to use. For more information about these guidelines, see Making requests in the *Amazon S3 API Reference*.

Using Amazon S3 Transfer Acceleration to accelerate geographically disparate data transfers

Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration is effective at minimizing or eliminating the latency caused by geographic distance between globally dispersed clients and a regional application using Amazon S3. Transfer Acceleration uses the globally distributed edge locations in CloudFront for data transport. The AWS edge network has points of presence in more than 50 locations. Today, it is used to distribute content through CloudFront and to provide rapid responses to DNS queries made to Amazon Route 53.

The edge network also helps to accelerate data transfers into and out of Amazon S3. It is ideal for applications that transfer data across or between continents, have a fast internet connection, use large objects, or have a lot of content to upload. As the data arrives at an edge location, data is routed to Amazon S3 over an optimized network path. In general, the farther away you are from an Amazon S3 Region, the higher the speed improvement you can expect from using Transfer Acceleration.

You can set up Transfer Acceleration on new or existing buckets. You can use a separate Amazon S3 Transfer Acceleration endpoint to use the AWS edge locations. The best way to test whether Transfer Acceleration helps client request performance is to use the Amazon S3 Transfer Acceleration Speed Comparison tool. Network configurations and conditions vary from time to time and from location to location. So you are charged only for transfers where Amazon S3 Transfer Acceleration can potentially improve your upload performance. For information about using Transfer Acceleration with different AWS SDKs, see Enabling and using S3 Transfer Acceleration.

Optimizing for high-request rate workloads

Applications that generate high request rates to Amazon S3 require specific design patterns to achieve optimal performance. When your application consistently generates more than 3,500 PUT/COPY/POST/DELETE or 5,500 GET/HEAD requests per second per prefix, you should implement strategies to distribute requests and handle scaling behavior.

Amazon S3 automatically scales to accommodate higher request rates, but this scaling happens gradually. During the scaling process, you might receive HTTP 503 (Slow Down) responses. These responses are temporary and indicate that Amazon S3 is optimizing its internal systems for your new request pattern. Once scaling is complete, your requests will be served without throttling.

To optimize performance for high-request rate workloads, consider the following strategies:

- **Distribute requests across multiple prefixes** – Use a randomized or sequential prefix pattern to spread requests across multiple partitions. For example, instead of using sequential object names like `log-2024-01-01.txt`, use randomized prefixes like `a1b2/log-2024-01-01.txt`. This helps Amazon S3 distribute the load more effectively.
- **Implement exponential backoff for 503 errors** – When you receive HTTP 503 responses, implement retry logic with exponential backoff. Start with a short delay and gradually increase the wait time between retries. The AWS SDKs include built-in retry logic that handles this automatically.
- **Monitor request patterns** – Use Amazon CloudWatch metrics to monitor your request rates and error rates. Pay particular attention to 5xx error metrics, which can indicate when your application is approaching or exceeding current scaling limits.
- **Gradually ramp up request rates** – When launching new applications or significantly increasing request rates, gradually increase your traffic over time rather than immediately jumping to peak rates. This allows Amazon S3 to scale proactively and reduces the likelihood of throttling.
- **Use multiple connections** – Distribute your requests across multiple HTTP connections to maximize throughput and reduce the impact of any single connection issues.

For applications that require consistent high performance, consider using Amazon S3 Express One Zone, which is designed for applications that require single-digit millisecond latencies and can support hundreds of thousands of requests per second. For more information, see S3 Express One Zone.

Hosting a static website using Amazon S3

You can use Amazon S3 to host a static website. On a *static* website, individual webpages include static content. They might also contain client-side scripts.

Note

We recommend that you use AWS Amplify Hosting to host static website content stored on S3. Amplify Hosting is a fully managed service that makes it easy to deploy your websites on a globally available content delivery network (CDN) powered by Amazon CloudFront, allowing secure static website hosting.
With AWS Amplify Hosting, you can select the location of your objects within your general purpose bucket, deploy your content to a managed CDN, and generate a public HTTPS URL for your website to be accessible anywhere. For more information about Amplify Hosting, see Deploying a static website to AWS Amplify Hosting from an S3 general purpose bucket and Deploying a static website from S3 using the Amplify console in the *AWS Amplify Console User Guide*.

For more information about hosting a static website on Amazon S3, including instructions and step-by-step walkthroughs, see the following topics.

Important

If the bucket that you're using to host your static website has been encrypted using server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), you must create an Amazon CloudFront distribution to serve your website because SSE-KMS doesn't support anonymous users. When you create your CloudFront distribution, you must use origin access control (OAC) instead of origin access identity (OAI) to secure the origin. OAI doesn't support SSE-KMS, so you must use OAC instead.
For more information about OAC, see Restrict access to an Amazon S3 origin in the *Amazon CloudFront Developer Guide*. For a tutorial that shows how to host a static website using Amazon CloudFront, see the section called “Hosting video streaming”.

Topics

- Website endpoints
- Enabling website hosting
- Configuring an index document
- Configuring a custom error document
- Setting permissions for website access
- (Optional) Logging web traffic
- (Optional) Configuring a webpage redirect
- Using cross-origin resource sharing (CORS)
- Static website tutorials

Website endpoints

When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. Website endpoints are different from the endpoints where you send REST API requests. For more information about the differences between the endpoints, see Key differences between a website endpoint and a REST API endpoint.

Depending on your Region, your Amazon S3 website endpoint follows one of these two formats.

- **s3-website dash (-) Region** - `http://bucket-name.s3-website-Region.amazonaws.com`
- **s3-website dot (.) Region** - `http://bucket-name.s3-website.Region.amazonaws.com`

These URLs return the default index document that you configure for the website. For a complete list of Amazon S3 website endpoints, see Amazon S3 Website Endpoints.

> **Note**
>
> To augment the security of your Amazon S3 static websites, the Amazon S3 website endpoint domains (for example, *s3-website-us-east-1.amazonaws.com* or *s3-website.ap-south-1.amazonaws.com*) are registered in the Public Suffix List (PSL). For further security, we recommend that you use cookies with a `__Host-` prefix if you ever need to set sensitive cookies in the domain name for your Amazon S3 static websites. This practice will help to defend your domain against cross-site request forgery attempts (CSRF). For more information see the Set-Cookie page in the Mozilla Developer Network.

If you want your website to be public, you must make all your content publicly readable for your customers to be able to access it at the website endpoint. For more information, see Setting permissions for website access.

Important

- Amazon S3 website endpoints do not support HTTPS or access points. If you want to use HTTPS, you can do one of the following:
  - (Recommended) Use AWS Amplify Hosting to host static website content stored on S3. Amplify Hosting is a fully managed service that makes it easy to deploy your websites on a globally available content delivery network (CDN) powered by Amazon CloudFront, allowing secure static website hosting.

    With AWS Amplify Hosting, you can select the location of your objects within your general purpose bucket, deploy your content to a managed CDN, and generate a public HTTPS URL for your website to be accessible anywhere. For more information about Amplify Hosting, see Deploying a static website to AWS Amplify Hosting from an S3 general purpose bucket and Deploying a static website from S3 using the Amplify console in the *AWS Amplify Console User Guide*.
  - Use Amazon CloudFront to serve a static website hosted on Amazon S3. For more information, see How do I use CloudFront to serve HTTPS requests for my Amazon S3 bucket? To use HTTPS with a custom domain, see Configuring a static website using a custom domain registered with Route 53.
- Requester Pays buckets do not allow access through a website endpoint. Any request to such a bucket receives a 403 Access Denied response. For more information, see Using Requester Pays general purpose buckets for storage transfers and usage.

Topics

- Website endpoint examples
- Adding a DNS CNAME
- Using a custom domain with Route 53
- Key differences between a website endpoint and a REST API endpoint

Website endpoint examples

The following examples show how you can access an Amazon S3 bucket that is configured as a static website.

Example— Requesting an object at the root level

To request a specific object that is stored at the root level in the bucket, use the following URL structure.

```
http://bucket-name.s3-website.Region.amazonaws.com/object-name
```

For example, the following URL requests the `photo.jpg` object that is stored at the root level in the bucket.

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/photo.jpg
```

Example— Requesting an object in a prefix

To request an object that is stored in a folder in your bucket, use this URL structure.

```
http://bucket-name.s3-website.Region.amazonaws.com/folder-name/object-name
```

The following URL requests the `docs/doc1.html` object in your bucket.

```
http://example-bucket.s3-website.us-west-2.amazonaws.com/docs/doc1.html
```

Adding a DNS CNAME

If you have a registered domain, you can add a DNS CNAME entry to point to the Amazon S3 website endpoint. For example, if you registered the domain `www.example-bucket.com`, you could create a bucket `www.example-bucket.com`, and add a DNS CNAME record that points to `www.example-bucket.com.s3-website.Region.amazonaws.com`. All requests to `http://www.example-bucket.com` are routed to `www.example-bucket.com.s3-website.Region.amazonaws.com`.

For more information, see Customizing Amazon S3 URLs with CNAME records.

Using a custom domain with Route 53

Instead of accessing the website using an Amazon S3 website endpoint, you can use your own domain registered with Amazon Route 53 to serve your content—for example, `example.com`. You can use Amazon S3 with Route 53 to host a website at the root domain. For example, if you have the root domain `example.com` and you host your website on Amazon S3, your website visitors can access the site from their browser by entering either `http://www.example.com` or `http://example.com`.

For an example walkthrough, see Tutorial: Configuring a static website using a custom domain registered with Route 53.

Key differences between a website endpoint and a REST API endpoint

An Amazon S3 website endpoint is optimized for access from a web browser. The following table summarizes the key differences between a REST API endpoint and a website endpoint.

Key difference	REST API endpoint	Website endpoint
Access control	Supports both public and private content	Supports only publicly readable content
Error message handling	Returns an XML-formatted error response	Returns an HTML document
Redirection support	Not applicable	Supports both object-level and bucket-level redirects
Requests supported	Supports all bucket and object operations	Supports only `GET` and `HEAD` requests on objects
Responses to `GET` and `HEAD` requests at the root of a bucket	Returns a list of the object keys in the bucket	Returns the index document that is specified in the website configuration

Key difference	REST API endpoint	Website endpoint
Secure Sockets Layer (SSL) support	Supports SSL connections	Does not support SSL connections

For a complete list of Amazon S3 endpoints, see Amazon S3 endpoints and quotas in the *AWS General Reference*.

Enabling website hosting

When you configure a bucket as a static website, you must enable static website hosting, configure an index document, and set permissions.

You can enable static website hosting using the Amazon S3 console, REST API, the AWS SDKs, the AWS CLI, or CloudFormation.

To configure your website with a custom domain, see Tutorial: Configuring a static website using a custom domain registered with Route 53.

Using the S3 console

To enable static website hosting

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to enable static website hosting for.
4. Choose **Properties**.
5. Under **Static website hosting**, choose **Edit**.
6. Choose **Use this bucket to host a website**.
7. Under **Static website hosting**, choose **Enable**.
8. In **Index document**, enter the file name of the index document, typically `index.html`.

   The index document name is case sensitive and must exactly match the file name of the HTML index document that you plan to upload to your S3 bucket. When you configure a bucket for

website hosting, you must specify an index document. Amazon S3 returns this index document when requests are made to the root domain or any of the subfolders. For more information, see Configuring an index document.

9. To provide your own custom error document for 4XX class errors, in **Error document**, enter the custom error document file name.

   The error document name is case sensitive and must exactly match the file name of the HTML error document that you plan to upload to your S3 bucket. If you don't specify a custom error document and an error occurs, Amazon S3 returns a default HTML error document. For more information, see Configuring a custom error document.

10. (Optional) If you want to specify advanced redirection rules, in **Redirection rules**, enter JSON to describe the rules.

    For example, you can conditionally route requests according to specific object key names or prefixes in the request. For more information, see Configure redirection rules to use advanced conditional redirects.

11. Choose **Save changes**.

    Amazon S3 enables static website hosting for your bucket. At the bottom of the page, under **Static website hosting**, you see the website endpoint for your bucket.

12. Under **Static website hosting**, note the **Endpoint**.

    The **Endpoint** is the Amazon S3 website endpoint for your bucket. After you finish configuring your bucket as a static website, you can use this endpoint to test your website.

Using the REST API

For more information about sending REST requests directly to enable static website hosting, see the following sections in the Amazon Simple Storage Service API Reference:

- PUT Bucket website
- GET Bucket website
- DELETE Bucket website

Using the AWS SDKs

To host a static website on Amazon S3, you configure an Amazon S3 bucket for website hosting and then upload your website content to the bucket. You can also use the AWS SDKs to create, update, and delete the website configuration programmatically. The SDKs provide wrapper classes around the Amazon S3 REST API. If your application requires it, you can send REST API requests directly from your application.

.NET

The following example shows how to use the AWS SDK for .NET to manage website configuration for a bucket. To add a website configuration to a bucket, you provide a bucket name and a website configuration. The website configuration must include an index document and can contain an optional error document. These documents must be stored in the bucket. For more information, see PUT Bucket website. For more information about the Amazon S3 website feature, see Hosting a static website using Amazon S3.

The following C# code example adds a website configuration to the specified bucket. The configuration specifies both the index document and the error document names. For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class WebsiteConfigTest
    {
        private const string bucketName = "*** bucket name ***";
        private const string indexDocumentSuffix = "*** index object key ***"; //
 For example, index.html.
        private const string errorDocument = "*** error object key ***"; // For
 example, error.html.
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 client;
```

```
        public static void Main()
        {
            client = new AmazonS3Client(bucketRegion);
            AddWebsiteConfigurationAsync(bucketName, indexDocumentSuffix,
errorDocument).Wait();
        }

        static async Task AddWebsiteConfigurationAsync(string bucketName,
                                               string indexDocumentSuffix,
                                               string errorDocument)
        {
            try
            {
                // 1. Put the website configuration.
                PutBucketWebsiteRequest putRequest = new PutBucketWebsiteRequest()
                {
                    BucketName = bucketName,
                    WebsiteConfiguration = new WebsiteConfiguration()
                    {
                        IndexDocumentSuffix = indexDocumentSuffix,
                        ErrorDocument = errorDocument
                    }
                };
                PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);

                // 2. Get the website configuration.
                GetBucketWebsiteRequest getRequest = new GetBucketWebsiteRequest()
                {
                    BucketName = bucketName
                };
                GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
                Console.WriteLine("Index document: {0}",
getResponse.WebsiteConfiguration.IndexDocumentSuffix);
                Console.WriteLine("Error document: {0}",
getResponse.WebsiteConfiguration.ErrorDocument);
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
```

```
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
 writing an object", e.Message);
            }
        }
    }
}
```

PHP

The following PHP example adds a website configuration to the specified bucket. The `create_website_config` method explicitly provides the index document and error document names. The example also retrieves the website configuration and prints the response. For more information about the Amazon S3 website feature, see Hosting a static website using Amazon S3.

For more information about the AWS SDK for Ruby API, go to AWS SDK for Ruby - Version 2.

```
 require 'vendor/autoload.php';

use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'version' => 'latest',
    'region'  => 'us-east-1'
]);


// Add the website configuration.
$s3->putBucketWebsite([
    'Bucket'                => $bucket,
    'WebsiteConfiguration'  => [
        'IndexDocument' => ['Suffix' => 'index.html'],
        'ErrorDocument' => ['Key' => 'error.html']
    ]
]);

// Retrieve the website configuration.
$result = $s3->getBucketWebsite([
    'Bucket' => $bucket
]);
```

```
echo $result->getPath('IndexDocument/Suffix');

// Delete the website configuration.
$s3->deleteBucketWebsite([
    'Bucket' => $bucket
]);
```

Using the AWS CLI

For more information about using the AWS CLI to configure an S3 bucket as a static website, see website in the *AWS CLI Command Reference*.

Next, you must configure your index document and set permissions. For information, see Configuring an index document and Setting permissions for website access.

You can also optionally configure an error document, web traffic logging, or a redirect.

Configuring an index document

When you enable website hosting, you must also configure and upload an index document. An *index document* is a webpage that Amazon S3 returns when a request is made to the root of a website or any subfolder. For example, if a user enters `http://www.example.com` in the browser, the user is not requesting any specific page. In that case, Amazon S3 serves up the index document, which is sometimes referred to as the *default page*.

When you enable static website hosting for your bucket, you enter the name of the index document (for example, `index.html`). After you enable static website hosting for your bucket, you upload an HTML file with the index document name to your bucket.

The trailing slash at the root-level URL is optional. For example, if you configure your website with `index.html` as the index document, either of the following URLs returns `index.html`.

```
http://example-bucket.s3-website.Region.amazonaws.com/
http://example-bucket.s3-website.Region.amazonaws.com
```

For more information about Amazon S3 website endpoints, see Website endpoints.

Index document and folders

In Amazon S3, a bucket is a flat container of objects. It does not provide any hierarchical organization as the file system on your computer does. However, you can create a logical hierarchy by using object key names that imply a folder structure.

For example, consider a bucket with three objects that have the following key names. Although these are stored with no physical hierarchical organization, you can infer the following logical folder structure from the key names:

- `sample1.jpg` — Object is at the root of the bucket.
- `photos/2006/Jan/sample2.jpg` — Object is in the `photos/2006/Jan` subfolder.
- `photos/2006/Feb/sample3.jpg` — Object is in the `photos/2006/Feb` subfolder.

In the Amazon S3 console, you can also create a folder in a bucket. For example, you can create a folder named `photos`. You can upload objects to the bucket or to the `photos` folder within the bucket. If you add the object `sample.jpg` to the bucket, the key name is `sample.jpg`. If you upload the object to the `photos` folder, the object key name is `photos/sample.jpg`.

If you create a folder structure in your bucket, you must have an index document at each level. In each folder, the index document must have the same name, for example, `index.html`. When a user specifies a URL that resembles a folder lookup, the presence or absence of a trailing slash determines the behavior of the website. For example, the following URL, with a trailing slash, returns the `photos/index.html` index document.

```
http://bucket-name.s3-website.Region.amazonaws.com/photos/
```

However, if you exclude the trailing slash from the preceding URL, Amazon S3 first looks for an object `photos` in the bucket. If the `photos` object is not found, it searches for an index document, `photos/index.html`. If that document is found, Amazon S3 returns a `302 Found` message and points to the `photos/` key. For subsequent requests to `photos/`, Amazon S3 returns `photos/index.html`. If the index document is not found, Amazon S3 returns an error.

Configure an index document

To configure an index document using the S3 console, use the following procedure. You can also configure an index document using the REST API, the AWS SDKs, the AWS CLI, or CloudFormation.

> **Note**
>
> In a versioning-enabled bucket, you may upload multiple copies of the `index.html` but only the newest version will be resolved to. For more information about using S3 Versioning see, Retaining multiple versions of objects with S3 Versioning.

When you enable static website hosting for your bucket, you enter the name of the index document (for example, **`index.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this index document name to your bucket.

To configure the index document

1. Create an `index.html` file.

   If you don't have an `index.html` file, you can use the following HTML to create one:

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" >
   <head>
       <title>My Website Home Page</title>
   </head>
   <body>
     <h1>Welcome to my website</h1>
     <p>Now hosted on Amazon S3!</p>
   </body>
   </html>
   ```

2. Save the index file locally.

   The index document file name must exactly match the index document name that you enter in the **Static website hosting** dialog box. The index document name is case sensitive. For example, if you enter `index.html` for the **Index document** name in the **Static website hosting** dialog box, your index document file name must also be `index.html` and not `Index.html`.

3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
4. In the left navigation pane, choose **General purpose buckets**.
5. In the buckets list, choose the name of the bucket that you want to use to host a static website.

6. Enable static website hosting for your bucket, and enter the exact name of your index document (for example, `index.html`). For more information, see Enabling website hosting.

   After enabling static website hosting, proceed to step 6.

7. To upload the index document to your bucket, do one of the following:

   - Drag and drop the index file into the console bucket listing.
   - Choose **Upload**, and follow the prompts to choose and upload the index file.

   For step-by-step instructions, see Uploading objects.

8. (Optional) Upload other website content to your bucket.

Next, you must set permissions for website access. For information, see Setting permissions for website access.

You can also optionally configure an error document, web traffic logging, or a redirect.

Configuring a custom error document

After you configure your bucket as a static website, when an error occurs, Amazon S3 returns an HTML error document. You can optionally configure your bucket with a custom error document so that Amazon S3 returns that document when an error occurs.

> **Note**
>
> Some browsers display their own error message when an error occurs, ignoring the error document that Amazon S3 returns. For example, when an HTTP 404 Not Found error occurs, Google Chrome might ignore the error document that Amazon S3 returns and display its own error.

Topics

- Amazon S3 HTTP response codes
- Configuring a custom error document

Amazon S3 HTTP response codes

The following table lists the subset of HTTP response codes that Amazon S3 returns when an error occurs.

HTTP error code	Description
301 Moved Permanently	When a user sends a request directly to the Amazon S3 website endpoint (`http://s3-website.` *Region*`.amazonaws.com/`), Amazon S3 returns a **301 Moved Permanently** response and redirects those requests to `https://aws.amazon.com/s3/`.
302 Found	When Amazon S3 receives a request for a key `x`, `http://`*bucket-name*`.s3-website.` *Region*`.amazonaws.com/x`, without a trailing slash, it first looks for the object with the key name `x`. If the object is not found, Amazon S3 determines that the request is for subfolder `x` and redirects the request by adding a slash at the end, and returns **302 Found**.
304 Not Modified	Amazon S3 uses request headers `If-Modified-Since`, `If-Unmodified-Since`, `If-Match` and/or `If-None-Match` to determine whether the requested object is same as the cached copy held by the client. If the object is the same, the website endpoint returns a **304 Not Modified** response.
400 Malformed Request	The website endpoint responds with a **400 Malformed Request** when a user attempts to access a bucket through the incorrect regional endpoint.
403 Forbidden	The website endpoint responds with a **403 Forbidden** when a user request translates to an object that is not publicly readable. The object owner must make the object publicly readable using a bucket policy or an ACL.
404 Not Found	The website endpoint responds with **404 Not Found** for the following reasons: •

HTTP error code	Description
	Amazon S3 determines that the URL of the website refers to an object key that does not exist. • Amazon S3 infers that the request is for an index document that does not exist. • A bucket specified in the URL does not exist. • A bucket specified in the URL exists, but isn't configured as a website. You can create a custom document that is returned for **404 Not Found**. Make sure that the document is uploaded to the bucket configured as a website, and that the website hosting configuration is set to use the document. For information on how Amazon S3 interprets the URL as a request for an object or an index document, see Configuring an index document.
500 Service Error	The website endpoint responds with a **500 Service Error** when an internal server error occurs.
503 Service Unavailable	The website endpoint responds with a **503 Service Unavailable** when Amazon S3 determines that you need to reduce your request rate.

For each of these errors, Amazon S3 returns a predefined HTML message. The following is an example HTML message that is returned for a **403 Forbidden** response.



Configuring a custom error document

When you configure your bucket as a static website, you can provide a custom error document that contains a user-friendly error message and additional help. Amazon S3 returns your custom error document for only the HTTP 4XX class of error codes.

To configure a custom error document using the S3 console, follow the steps below. You can also configure an error document using the REST API, the AWS SDKs, the AWS CLI, or CloudFormation. For more information, see the following:

- PutBucketWebsite in the *Amazon Simple Storage Service API Reference*
- AWS::S3::Bucket WebsiteConfiguration in the *CloudFormation User Guide*
- put-bucket-website in the *AWS CLI Command Reference*

When you enable static website hosting for your bucket, you enter the name of the error document (for example, **`404.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this error document name to your bucket.

To configure an error document

1. Create an error document, for example `404.html`.
2. Save the error document file locally.

   The error document name is case sensitive and must exactly match the name that you enter when you enable static website hosting. For example, if you enter `404.html` for the **Error**

document name in the **Static website hosting** dialog box, your error document file name must also be `404.html`.

3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
4. In the left navigation pane, choose **General purpose buckets**.
5. In the buckets list, choose the name of the bucket that you want to use to host a static website.
6. Enable static website hosting for your bucket, and enter the exact name of your error document (for example, `404.html`). For more information, see Enabling website hosting and Configuring a custom error document.

   After enabling static website hosting, proceed to step 6.
7. To upload the error document to your bucket, do one of the following:

   - Drag and drop the error document file into the console bucket listing.
   - Choose **Upload**, and follow the prompts to choose and upload the index file.

   For step-by-step instructions, see Uploading objects.

Setting permissions for website access

When you configure a bucket as a static website, if you want your website to be public, you can grant public read access. To make your bucket publicly readable, you must disable block public access settings for the bucket and write a bucket policy that grants public read access. If your bucket contains objects that are not owned by the bucket owner, you might also need to add an object access control list (ACL) that grants everyone read access.

If you don't want to disable block public access settings for your bucket but you still want your website to be public, you can create a Amazon CloudFront distribution to serve your static website. For more information, see Speeding up your website with Amazon CloudFront or Use an Amazon CloudFront distribution to serve a static website in the *Amazon Route 53 Developer Guide*.

> **Note**
>
> On the website endpoint, if a user requests an object that doesn't exist, Amazon S3 returns HTTP response code `404 (Not Found)`. If the object exists but you haven't granted

read permission on it, the website endpoint returns HTTP response code `403 (Access Denied)`. The user can use the response code to infer whether a specific object exists. If you don't want this behavior, you should not enable website support for your bucket.

Topics

- Step 1: Edit S3 Block Public Access settings
- Step 2: Add a bucket policy
- Object access control lists

Step 1: Edit S3 Block Public Access settings

If you want to configure an existing bucket as a static website that has public access, you must edit Block Public Access settings for that bucket. You might also have to edit your account-level Block Public Access settings. Amazon S3 applies the most restrictive combination of the bucket-level and account-level block public access settings.

For example, if you allow public access for a bucket but block all public access at the account level, Amazon S3 will continue to block public access to the bucket. In this scenario, you would have to edit your bucket-level and account-level Block Public Access settings. For more information, see Blocking public access to your Amazon S3 storage.

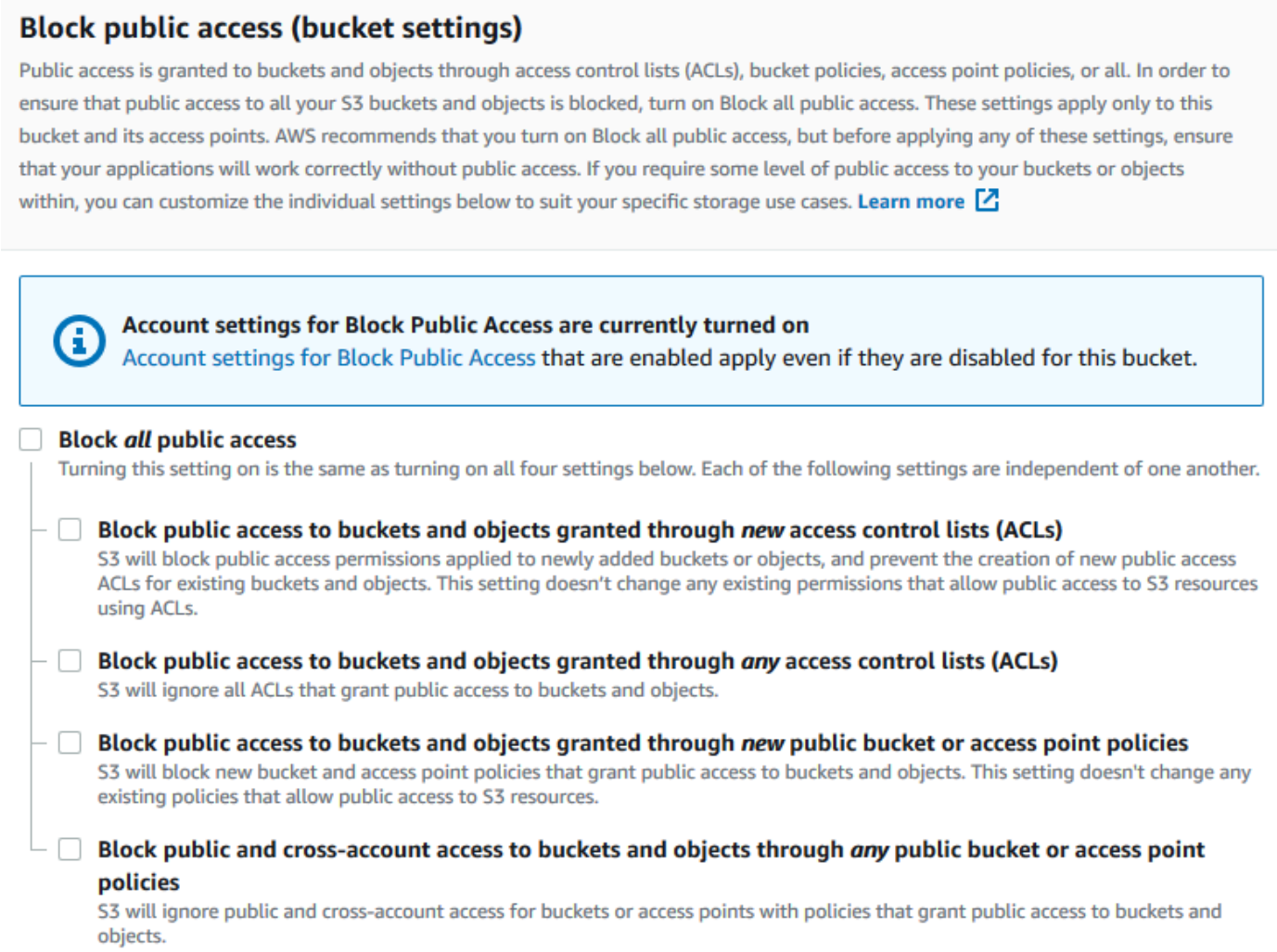
By default, Amazon S3 blocks public access to your account and buckets. If you want to use a bucket to host a static website, you can use these steps to edit your block public access settings.

Warning

Before you complete these steps, review Blocking public access to your Amazon S3 storage to ensure that you understand and accept the risks involved with allowing public access. When you turn off block public access settings to make your bucket public, anyone on the internet can access your bucket. We recommend that you block all public access to your buckets.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the bucket that you have configured as a static website.

3. Choose **Permissions**.
4. Under **Block public access (bucket settings)**, choose **Edit**.
5. Clear **Block *all* public access**, and choose **Save changes**.



Amazon S3 turns off the Block Public Access settings for your bucket. To create a public static website, you might also have to edit the Block Public Access settings for your account before adding a bucket policy. If the Block Public Access settings for your account are currently turned on, you see a note under **Block public access (bucket settings)**.

Step 2: Add a bucket policy

To make the objects in your bucket publicly readable, you must write a bucket policy that grants everyone `s3:GetObject` permission.

After you edit S3 Block Public Access settings, you can add a bucket policy to grant public read access to your bucket. When you grant public read access, anyone on the internet can access your bucket.

> **Important**
>
> The following policy is an example only and allows full access to the contents of your bucket. Before you proceed with this step, review How can I secure the files in my Amazon S3 bucket? to ensure that you understand the best practices for securing the files in your S3 bucket and risks involved in granting public access.

1. Under **Buckets**, choose the name of your bucket.
2. Choose **Permissions**.
3. Under **Bucket Policy**, choose **Edit**.
4. To grant public read access for your website, copy the following bucket policy, and paste it in the **Bucket policy editor**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. Update the `Resource` to your bucket name.

   In the preceding example bucket policy, *`Bucket-Name`* is a placeholder for the bucket name. To use this bucket policy with your own bucket, you must update this name to match your bucket name.

6. Choose **Save changes**.

   A message appears indicating that the bucket policy has been successfully added.

   If you see an error that says `Policy has invalid resource`, confirm that the bucket name in the bucket policy matches your bucket name. For information about adding a bucket policy, see How do I add an S3 bucket policy?

   If you get an error message and cannot save the bucket policy, check your account and bucket Block Public Access settings to confirm that you allow public access to the bucket.

Object access control lists

You can use a bucket policy to grant public read permission to your objects. However, the bucket policy applies only to objects that are owned by the bucket owner. If your bucket contains objects that aren't owned by the bucket owner, the bucket owner should use the object access control list (ACL) to grant public READ permission on those objects.

S3 Object Ownership is an Amazon S3 bucket-level setting that you can use to both control ownership of the objects that are uploaded to your bucket and to disable or enable ACLs. By default, Object Ownership is set to the Bucket owner enforced setting, and all ACLs are disabled. When ACLs are disabled, the bucket owner owns all the objects in the bucket and manages access to them exclusively by using access-management policies.

A majority of modern use cases in Amazon S3 no longer require the use of ACLs. We recommend that you keep ACLs disabled, except in circumstances where you need to control access for each object individually. With ACLs disabled, you can use policies to control access to all objects in your bucket, regardless of who uploaded the objects to your bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

> **Important**
>
> If your general purpose bucket uses the Bucket owner enforced setting for S3 Object Ownership, you must use policies to grant access to your general purpose bucket and the objects in it. With the Bucket owner enforced setting enabled, requests to set access control lists (ACLs) or update ACLs fail and return the `AccessControlListNotSupported` error code. Requests to read ACLs are still supported.

To make an object publicly readable using an ACL, grant READ permission to the `AllUsers` group, as shown in the following grant element. Add this grant element to the object ACL. For information about managing ACLs, see Access control list (ACL) overview.

```
<Grant>
  <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
```

(Optional) Logging web traffic

You can optionally enable Amazon S3 server access logging for a bucket that is configured as a static website. Server access logging provides detailed records for the requests that are made to your bucket. For more information, see Logging requests with server access logging. If you plan to use Amazon CloudFront to speed up your website, you can also use CloudFront logging. For more information, see Configuring and Using Access Logs in the *Amazon CloudFront Developer Guide*.

To enable server access logging for your static website bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the same Region where you created the bucket that is configured as a static website, create a general purpose bucket for logging, for example `logs.example.com`.
3. Create a folder for the server access logging log files (for example, `logs`).
4. (Optional) If you want to use CloudFront to improve your website performance, create a folder for the CloudFront log files (for example, `cdn`).

   For more information, see Speeding up your website with Amazon CloudFront.
5. In the **Buckets** list, choose your bucket.
6. Choose **Properties**.
7. Under **Server access logging**, choose **Edit**.
8. Choose **Enable**.
9. Under the **Target bucket**, choose the bucket and folder destination for the server access logs:

   - Browse to the folder and bucket location:

1. Choose **Browse S3**.
2. Choose the bucket name, and then choose the logs folder.
3. Choose **Choose path**.

- Enter the S3 bucket path, for example, **`s3://logs.example.com/logs/`**.

10. Choose **Save changes**.

    In your log bucket, you can now access your logs. Amazon S3 writes website access logs to your log bucket every 2 hours.

(Optional) Configuring a webpage redirect

If your Amazon S3 bucket is configured for static website hosting, you can configure redirects for your bucket or the objects in it. You have the following options for configuring redirects.

Topics

- Redirect requests for your bucket's website endpoint to another bucket or domain
- Configure redirection rules to use advanced conditional redirects
- Redirect requests for an object

Redirect requests for your bucket's website endpoint to another bucket or domain

You can redirect all requests to a website endpoint for a bucket to another bucket or domain. If you redirect all requests, any request made to the website endpoint is redirected to the specified bucket or domain.

For example, if your root domain is `example.com`, and you want to serve requests for both `http://example.com` and `http://www.example.com`, you must create two buckets named `example.com` and `www.example.com`. Then, maintain the content in the `example.com` bucket, and configure the other `www.example.com` bucket to redirect all requests to the `example.com` bucket. For more information, see Configuring a Static Website Using a Custom Domain Name.

To redirect requests for a bucket website endpoint

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. Under **Buckets**, choose the name of the bucket that you want to redirect requests from (for example, `www.example.com`).
3. Choose **Properties**.
4. Under **Static website hosting**, choose **Edit**.
5. Choose **Redirect requests for an object**.
6. In the **Host name** box, enter the website endpoint for your bucket or your custom domain.

   For example, if you are redirecting to a root domain address, you would enter **`example.com`**.
7. For **Protocol**, choose the protocol for the redirected requests (**none**,**http**, or **https**).

   If you do not specify a protocol, the default option is **none**.
8. Choose **Save changes**.

Configure redirection rules to use advanced conditional redirects

Using advanced redirection rules, you can route requests conditionally according to specific object key names, prefixes in the request, or response codes. For example, suppose that you delete or rename an object in your bucket. You can add a routing rule that redirects the request to another object. If you want to make a folder unavailable, you can add a routing rule to redirect the request to another webpage. You can also add a routing rule to handle error conditions by routing requests that return the error to another domain when the error is processed.

When enabling static website hosting for your bucket, you can optionally specify advanced redirection rules. Amazon S3 has a limitation of 50 routing rules per website configuration. If you require more than 50 routing rules, you can use object redirect. For more information, see Using the S3 console.

For more information about configuring routing rules using the REST API, see PutBucketWebsite in the *Amazon Simple Storage Service API Reference*.

> **Important**
>
> To create redirection rules in the new Amazon S3 console, you must use JSON. For JSON examples, see Redirection rules examples.

To configure redirection rules for a static website

To add redirection rules for a bucket that already has static website hosting enabled, follow these steps.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of a bucket that you have configured as a static website.
4. Choose **Properties**.
5. Under **Static website hosting**, choose **Edit**.
6. In **Redirection rules** box, enter your redirection rules in JSON.

   In the S3 console you describe the rules using JSON. For JSON examples, see Redirection rules examples. Amazon S3 has a limitation of 50 routing rules per website configuration.
7. Choose **Save changes**.

Routing rule elements

The following is general syntax for defining the routing rules in a website configuration in JSON and XML To configure redirection rules in the new S3 console, you must use JSON. For JSON examples, see Redirection rules examples.

JSON

```
[
    {
      "Condition": {
        "HttpErrorCodeReturnedEquals": "string",
        "KeyPrefixEquals": "string"
      },
      "Redirect": {
        "HostName": "string",
        "HttpRedirectCode": "string",
        "Protocol": "http"|"https",
        "ReplaceKeyPrefixWith": "string",
        "ReplaceKeyWith": "string"
      }
    }
  ]
```

```
Note: Redirect must each have at least one child element. You can have either
 ReplaceKeyPrefix with or ReplaceKeyWith but not both.
```

XML

```
<RoutingRules> =
    <RoutingRules>
         <RoutingRule>...</RoutingRule>
         [<RoutingRule>...</RoutingRule>
         ...]
    </RoutingRules>

<RoutingRule> =
   <RoutingRule>
      [ <Condition>...</Condition> ]
      <Redirect>...</Redirect>
   </RoutingRule>

<Condition> =
   <Condition>
      [ <KeyPrefixEquals>...</KeyPrefixEquals> ]
      [ <HttpErrorCodeReturnedEquals>...</HttpErrorCodeReturnedEquals> ]
   </Condition>
    Note: <Condition> must have at least one child element.

<Redirect> =
   <Redirect>
      [ <HostName>...</HostName> ]
      [ <Protocol>...</Protocol> ]
      [ <ReplaceKeyPrefixWith>...</ReplaceKeyPrefixWith>  ]
      [ <ReplaceKeyWith>...</ReplaceKeyWith> ]
      [ <HttpRedirectCode>...</HttpRedirectCode> ]
   </Redirect>

Note: <Redirect> must have at least one child element. You can have either
 ReplaceKeyPrefix with or ReplaceKeyWith but not both.
```

The following table describes the elements in the routing rule.

Name	Description
RoutingRules	Container for a collection of RoutingRule elements.
RoutingRule	A rule that identifies a condition and the redirect that is applied when the condition is met. Condition: • A RoutingRules container must contain at least one routing rule.
Condition	Container for describing a condition that must be met for the specified redirect to be applied. If the routing rule does not include a condition, the rule is applied to all requests.
KeyPrefixEquals	The prefix of the object key name from which requests are redirected. KeyPrefixEquals is required if HttpErrorCodeReturnedEquals is not specified. If both KeyPrefixEquals and HttpErrorCodeReturnedEquals are specified, both must be true for the condition to be met.
HttpErrorCodeReturnedEquals	The HTTP error code that must match for the redirect to apply. If an error occurs, and if the error code meets this value, then the specified redirect applies. HttpErrorCodeReturnedEquals is required if KeyPrefixEquals is not specified. If both KeyPrefixEquals and HttpErrorCodeReturnedEquals are specified, both must be true for the condition to be met.
Redirect	

Name	Description
	Container element that provides instructions for redirecting the request. You can redirect requests to another host or another page, or you can specify another protocol to use. A `RoutingRule` must have a `Redirect` element. A `Redirect` element must contain at least one of the following sibling elements: `Protocol`, `HostName`, `ReplaceKeyPrefixWith`, `ReplaceKeyWith`, or `HttpRedirectCode`.
`Protocol`	The protocol, `http` or `https`, to be used in the `Location` header that is returned in the response. If one of its siblings is supplied, `Protocol` is not required.
`HostName`	The hostname to be used in the `Location` header that is returned in the response. If one of its siblings is supplied, `HostName` is not required.
`ReplaceKeyPrefixWith`	The prefix of the object key name that replaces the value of `KeyPrefixEquals` in the redirect request. If one of its siblings is supplied, `ReplaceKeyPrefixWith` is not required. It can be supplied only if `ReplaceKeyWith` is not supplied.
`ReplaceKeyWith`	The object key to be used in the `Location` header that is returned in the response. If one of its siblings is supplied, `ReplaceKeyWith` is not required. It can be supplied only if `ReplaceKeyPrefixWith` is not supplied.

Name	Description
`HttpRedirectCode`	The HTTP redirect code to be used in the `Location` header that is returned in the response. If one of its siblings is supplied, `HttpRedirectCode` is not required.

Redirection rules examples

The following examples explain common redirection tasks:

> **Important**
>
> To create redirection rules in the new Amazon S3 console, you must use JSON.

Example 1: Redirect after renaming a key prefix

Suppose that your bucket contains the following objects:

- index.html
- docs/article1.html
- docs/article2.html

You decide to rename the folder from `docs/` to `documents/`. After you make this change, you need to redirect requests for prefix `docs/` to `documents/`. For example, request for `docs/article1.html` will be redirected to `documents/article1.html`.

In this case, you add the following routing rule to the website configuration.

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "docs/"
        },
```

```
        "Redirect": {
            "ReplaceKeyPrefixWith": "documents/"
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
    <Condition>
      <KeyPrefixEquals>docs/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyPrefixWith>documents/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

Example 2: Redirect requests for a deleted folder to a page

Suppose that you delete the `images/` folder (that is, you delete all objects with the key prefix `images/`). You can add a routing rule that redirects requests for any object with the key prefix `images/` to a page named `folderdeleted.html`.

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "images/"
        },
        "Redirect": {
            "ReplaceKeyWith": "folderdeleted.html"
        }
    }
]
```

XML

```
  <RoutingRules>
```

```
    <RoutingRule>
    <Condition>
       <KeyPrefixEquals>images/</KeyPrefixEquals>
    </Condition>
    <Redirect>
      <ReplaceKeyWith>folderdeleted.html</ReplaceKeyWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

Example 3: Redirect to another domain with a specific path

Suppose you want to redirect requests for a specific path to another domain. For example, you want to redirect requests for `/redirect/me` to `https://example.com/new/path`.

When using both `HostName` and `ReplaceKeyWith` together, Amazon S3 constructs the redirect URL by concatenating the hostname and the replacement key with a forward slash between them. Therefore, you should not include a leading slash in the `ReplaceKeyWith` value. Amazon S3 automatically adds the forward slash between the hostname and the replacement key.

JSON

```
[
    {
        "Condition": {
            "KeyPrefixEquals": "redirect/me"
        },
        "Redirect": {
            "HostName": "example.com",
            "ReplaceKeyWith": "new/path"
        }
    }
]
```

XML

```
  <RoutingRules>
    <RoutingRule>
    <Condition>
      <KeyPrefixEquals>redirect/me</KeyPrefixEquals>
    </Condition>
```

```
    <Redirect>
      <HostName>example.com</HostName>
      <ReplaceKeyWith>new/path</ReplaceKeyWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

This configuration redirects a request for `https://yourbucket.s3-website-region.amazonaws.com/redirect/me` to `https://example.com/new/path`. Note that `ReplaceKeyWith` is set to `new/path` without a leading slash.

Example 4: Redirect for an HTTP error

Suppose that when a requested object is not found, you want to redirect requests to an Amazon Elastic Compute Cloud (Amazon EC2) instance. Add a redirection rule so that when an HTTP status code 404 (Not Found) is returned, the site visitor is redirected to an Amazon EC2 instance that handles the request.

The following example also inserts the object key prefix `report-404/` in the redirect. For example, if you request a page `ExamplePage.html` and it results in an HTTP 404 error, the request is redirected to a page `report-404/ExamplePage.html` on the specified Amazon EC2 instance. If there is no routing rule and the HTTP error 404 occurs, the error document that is specified in the configuration is returned.

JSON

```
[
    {
        "Condition": {
            "HttpErrorCodeReturnedEquals": "404"
        },
        "Redirect": {
            "HostName": "ec2-11-22-333-44.compute-1.amazonaws.com",
            "ReplaceKeyPrefixWith": "report-404/"
        }
    }
]
```

XML

```
  <RoutingRules>
```

```
    <RoutingRule>
    <Condition>
      <HttpErrorCodeReturnedEquals>404</HttpErrorCodeReturnedEquals >
    </Condition>
    <Redirect>
      <HostName>ec2-11-22-333-44.compute-1.amazonaws.com</HostName>
      <ReplaceKeyPrefixWith>report-404/</ReplaceKeyPrefixWith>
    </Redirect>
    </RoutingRule>
  </RoutingRules>
```

Redirect requests for an object

You can redirect requests for an object to another object or URL by setting the website redirect location in the metadata of the object. You set the redirect by adding the `x-amz-website-redirect-location` property to the object metadata. On the Amazon S3 console, you set the **Website Redirect Location** in the metadata of the object. If you use the Amazon S3 API, you set `x-amz-website-redirect-location`. The website then interprets the object as a 301 redirect.

To redirect a request to another object, you set the redirect location to the key of the target object. To redirect a request to an external URL, you set the redirect location to the URL that you want. For more information about object metadata, see System-defined object metadata.

When you set a page redirect, you can either keep or delete the source object content. For example, if you have a `page1.html` object in your bucket, you can redirect any requests for this page to another object, `page2.html`. You have two options:

- Keep the content of the `page1.html` object and redirect page requests.
- Delete the content of `page1.html` and upload a zero-byte object named `page1.html` to replace the existing object and redirect page requests.

Using the S3 console

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the **Buckets** list, choose the name of the bucket that you have configured as a static website (for example, `example.com`).
3. Under **Objects**, select your object.
4. Choose **Actions**, and choose **Edit metadata**.

5. Choose **Metadata**.
6. Choose **Add Metadata**.
7. Under **Type**, choose **System Defined**.
8. In **Key**, choose **x-amz-website-redirect-location**.
9. In **Value**, enter the key name of the object that you want to redirect to, for example, `/page2.html`.

   For another object in the same bucket, the `/` prefix in the value is required. You can also set the value to an external URL, for example, `http://www.example.com`.
10. Choose **Edit metadata**.

Using the REST API

The following Amazon S3 API actions support the `x-amz-website-redirect-location` header in the request. Amazon S3 stores the header value in the object metadata as `x-amz-website-redirect-location`.

- PUT Object
- Initiate Multipart Upload
- POST Object
- PUT Object - Copy

A bucket configured for website hosting has both the website endpoint and the REST endpoint. A request for a page that is configured as a 301 redirect has the following possible outcomes, depending on the endpoint of the request:

- **Region-specific website endpoint –** Amazon S3 redirects the page request according to the value of the `x-amz-website-redirect-location` property.
- **REST endpoint –** Amazon S3 doesn't redirect the page request. It returns the requested object.

For more information about the endpoints, see Key differences between a website endpoint and a REST API endpoint.

When setting a page redirect, you can either keep or delete the object content. For example, suppose that you have a `page1.html` object in your bucket.

- To keep the content of `page1.html` and only redirect page requests, you can submit a PUT Object - Copy request to create a new `page1.html` object that uses the existing `page1.html` object as the source. In your request, you set the `x-amz-website-redirect-location` header. When the request is complete, you have the original page with its content unchanged, but Amazon S3 redirects any requests for the page to the redirect location that you specify.
- To delete the content of the `page1.html` object and redirect requests for the page, you can send a PUT Object request to upload a zero-byte object that has the same object key: `page1.html`. In the PUT request, you set `x-amz-website-redirect-location` for `page1.html` to the new object. When the request is complete, `page1.html` has no content, and requests are redirected to the location that is specified by `x-amz-website-redirect-location`.

When you retrieve the object using the GET Object action, along with other object metadata, Amazon S3 returns the `x-amz-website-redirect-location` header in the response.

Using cross-origin resource sharing (CORS)

Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain. With CORS support, you can build rich client-side web applications with Amazon S3 and selectively allow cross-origin access to your Amazon S3 resources.

This section provides an overview of CORS. The subtopics describe how you can enable CORS using the Amazon S3 console, or programmatically by using the Amazon S3 REST API and the AWS SDKs.

Cross-origin resource sharing: Use-case scenarios

The following are example scenarios for using CORS.

Scenario 1

Suppose that you are hosting a website in an Amazon S3 bucket named `website` as described in Hosting a static website using Amazon S3. Your users load the website endpoint:

```
http://website.s3-website.us-east-1.amazonaws.com
```

Now you want to use JavaScript on the webpages that are stored in this bucket to be able to make authenticated GET and PUT requests against the same bucket by using the Amazon S3

API endpoint for the bucket, `website.s3.us-east-1.amazonaws.com`. A browser would normally block JavaScript from allowing those requests, but with CORS you can configure your bucket to explicitly enable cross-origin requests from `website.s3-website.us-east-1.amazonaws.com`.

Scenario 2

Suppose that you want to host a web font from your S3 bucket. Again, browsers require a CORS check (also called a preflight check) for loading web fonts. You would configure the bucket that is hosting the web font to allow any origin to make these requests.

How does Amazon S3 evaluate the CORS configuration on a bucket?

When Amazon S3 receives a preflight request from a browser, it evaluates the CORS configuration for the bucket and uses the first `CORSRule` rule that matches the incoming browser request to enable a cross-origin request. For a rule to match, the following conditions must be met:

- The `Origin` header in a CORS request to your bucket must match the origins in the `AllowedOrigins` element in your CORS configuration.
- The HTTP methods that are specified in the `Access-Control-Request-Method` in a CORS request to your bucket must match the method or methods listed in the `AllowedMethods` element in your CORS configuration.
- The headers listed in the `Access-Control-Request-Headers` header in a pre-flight request must match the headers in the `AllowedHeaders` element in your CORS configuration.

> **Note**
>
> The ACLs and policies continue to apply when you enable CORS on your bucket.

How Object Lambda Access Point supports CORS

When S3 Object Lambda receives a request from a browser or the request includes an `Origin` header, S3 Object Lambda always adds an `"AllowedOrigins":"*"` header field.

For more information about using CORS, see the following topics.

Topics

- Elements of a CORS configuration
- Configuring cross-origin resource sharing (CORS)
- Testing CORS
- Troubleshooting CORS

Elements of a CORS configuration

To configure your bucket to allow cross-origin requests, you create a CORS configuration. The CORS configuration is a document with elements that identify the origins that you will allow to access your bucket, the operations (HTTP methods) that you will support for each origin, and other operation-specific information. You can add up to 100 rules to the configuration. You can add the CORS configuration as the `cors` subresource to the bucket.

If you are configuring CORS in the S3 console, you must use JSON to create a CORS configuration. The new S3 console only supports JSON CORS configurations.

For more information about the CORS configuration and the elements in it, see the topics below. For instructions on how to add a CORS configuration, see Configuring cross-origin resource sharing (CORS).

> **Important**
>
> In the S3 console, the CORS configuration must be JSON.

Topics

- AllowedMethods element
- AllowedOrigins element
- AllowedHeaders element
- ExposeHeaders element
- MaxAgeSeconds element
- Examples of CORS configurations

`AllowedMethods` element

In the CORS configuration, you can specify the following values for the `AllowedMethods` element.

- GET
- PUT
- POST
- DELETE
- HEAD

`AllowedOrigins` element

In the `AllowedOrigins` element, you specify the origins that you want to allow cross-domain requests from, for example, `http://www.example.com`. The origin string can contain only one `*` wildcard character, such as `http://*.example.com`. You can optionally specify `*` as the origin to enable all the origins to send cross-origin requests. You can also specify `https` to enable only secure origins.

`AllowedHeaders` element

The `AllowedHeaders` element specifies which headers are allowed in a preflight request through the `Access-Control-Request-Headers` header. Each header name in the `Access-Control-Request-Headers` header must match a corresponding entry in the element. Amazon S3 will send only the allowed headers in a response that were requested. For a sample list of headers that can be used in requests to Amazon S3, go to Common Request Headers in the *Amazon Simple Storage Service API Reference* guide.

Each AllowedHeaders string in your configuration can contain at most one * wildcard character. For example, `<AllowedHeader>x-amz-*</AllowedHeader>` will enable all Amazon-specific headers.

`ExposeHeaders` element

Each `ExposeHeader` element identifies a header in the response that you want customers to be able to access from their applications (for example, from a JavaScript `XMLHttpRequest` object). For a list of common Amazon S3 response headers, go to Common Response Headers in the *Amazon Simple Storage Service API Reference* guide.

`MaxAgeSeconds` element

The `MaxAgeSeconds` element specifies the time in seconds that your browser can cache the response for a preflight request as identified by the resource, the HTTP method, and the origin.

Examples of CORS configurations

Instead of accessing a website by using an Amazon S3 website endpoint, you can use your own domain, such as `example1.com` to serve your content. For information about using your own domain, see Tutorial: Configuring a static website using a custom domain registered with Route 53.

The following example CORS configuration has three rules, which are specified as `CORSRule` elements:

- The first rule allows cross-origin PUT, POST, and DELETE requests from the `http://www.example1.com` origin. The rule also allows all headers in a preflight OPTIONS request through the `Access-Control-Request-Headers` header. In response to preflight OPTIONS requests, Amazon S3 returns requested headers.
- The second rule allows the same cross-origin requests as the first rule, but the rule applies to another origin, `http://www.example2.com`.
- The third rule allows cross-origin GET requests from all origins. The `*` wildcard character refers to all origins.

JSON

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example1.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
```

```
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example2.com"
        ],
        "ExposeHeaders": []
    },
    {
        "AllowedHeaders": [],
        "AllowedMethods": [
            "GET"
        ],
        "AllowedOrigins": [
            "*"
        ],
        "ExposeHeaders": []
    }
]
```

XML

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example1.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
   <AllowedOrigin>http://www.example2.com</AllowedOrigin>

   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>

   <AllowedHeader>*</AllowedHeader>
 </CORSRule>
 <CORSRule>
```

```
   <AllowedOrigin>*</AllowedOrigin>
   <AllowedMethod>GET</AllowedMethod>
 </CORSRule>
</CORSConfiguration>
```

The CORS configuration also allows optional configuration parameters, as shown in the following CORS configuration. In this example, the CORS configuration allows cross-origin PUT, POST, and DELETE requests from the `http://www.example.com` origin.

JSON

```
[
    {
        "AllowedHeaders": [
            "*"
        ],
        "AllowedMethods": [
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example.com"
        ],
        "ExposeHeaders": [
            "x-amz-server-side-encryption",
            "x-amz-request-id",
            "x-amz-id-2"
        ],
        "MaxAgeSeconds": 3000
    }
]
```

XML

```
<CORSConfiguration>
 <CORSRule>
   <AllowedOrigin>http://www.example.com</AllowedOrigin>
   <AllowedMethod>PUT</AllowedMethod>
   <AllowedMethod>POST</AllowedMethod>
   <AllowedMethod>DELETE</AllowedMethod>
   <AllowedHeader>*</AllowedHeader>
```

```
  <MaxAgeSeconds>3000</MaxAgeSeconds>
  <ExposeHeader>x-amz-server-side-encryption</
ExposeHeader>
  <ExposeHeader>x-amz-request-id</
ExposeHeader>
  <ExposeHeader>x-amz-id-2</ExposeHeader>
 </CORSRule>
</CORSConfiguration>
```

The `CORSRule` element in the preceding configuration includes the following optional elements:

- `MaxAgeSeconds`—Specifies the amount of time in seconds (in this example, 3000) that the browser caches an Amazon S3 response to a preflight OPTIONS request for the specified resource. By caching the response, the browser does not have to send preflight requests to Amazon S3 if the original request will be repeated.
- `ExposeHeaders`—Identifies the response headers (in this example, `x-amz-server-side-encryption`, `x-amz-request-id`, and `x-amz-id-2`) that customers are able to access from their applications (for example, from a JavaScript `XMLHttpRequest` object).

Configuring cross-origin resource sharing (CORS)

Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain. With CORS support, you can build rich client-side web applications with Amazon S3 and selectively allow cross-origin access to your Amazon S3 resources.

This section shows you how to enable CORS using the Amazon S3 console, the Amazon S3 REST API, and the AWS SDKs. To configure your bucket to allow cross-origin requests, you add a CORS configuration to the bucket. A CORS configuration is a document that defines rules that identify the origins that you will allow to access your bucket, the operations (HTTP methods) supported for each origin, and other operation-specific information. In the S3 console, the CORS configuration must be a JSON document.

For example CORS configurations in JSON and XML, see Elements of a CORS configuration.

Using the S3 console

This section explains how to use the Amazon S3 console to add a cross-origin resource sharing (CORS) configuration to an S3 bucket.

When you enable CORS on the bucket, the access control lists (ACLs) and other access permission policies continue to apply.

> **Important**
>
> In the S3 console, the CORS configuration must be JSON. For examples CORS configurations in JSON and XML, see Elements of a CORS configuration.

To add a CORS configuration to an S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to create a bucket policy for.
4. Choose **Permissions**.
5. In the **Cross-origin resource sharing (CORS)** section, choose **Edit**.
6. In the **CORS configuration editor** text box, type or copy and paste a new CORS configuration, or edit an existing configuration.

   The CORS configuration is a JSON file. The text that you type in the editor must be valid JSON. For more information, see Elements of a CORS configuration.
7. Choose **Save changes**.

   > **Note**
   >
   > Amazon S3 displays the Amazon Resource Name (ARN) for the bucket next to the **CORS configuration editor** title. For more information about ARNs, see Amazon Resource Names (ARNs) and AWS Service Namespaces in the *Amazon Web Services General Reference*.

Using the AWS SDKs

You can use the AWS SDK to manage cross-origin resource sharing (CORS) for a bucket. For more information about CORS, see Using cross-origin resource sharing (CORS).

The following examples:

- Creates a CORS configuration and sets the configuration on a bucket
- Retrieves the configuration and modifies it by adding a rule
- Adds the modified configuration to the bucket
- Deletes the configuration

Java

Example

Example

For instructions on how to create and test a working sample, see Getting Started in the AWS SDK for Java Developer Guide.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.BucketCrossOriginConfiguration;
import com.amazonaws.services.s3.model.CORSRule;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class CORS {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        // Create two CORS rules.
        List<CORSRule.AllowedMethods> rule1AM = new
 ArrayList<CORSRule.AllowedMethods>();
        rule1AM.add(CORSRule.AllowedMethods.PUT);
        rule1AM.add(CORSRule.AllowedMethods.POST);
        rule1AM.add(CORSRule.AllowedMethods.DELETE);
```

```
        CORSRule rule1 = new
CORSRule().withId("CORSRule1").withAllowedMethods(rule1AM)
                .withAllowedOrigins(Arrays.asList("http://*.example.com"));

        List<CORSRule.AllowedMethods> rule2AM = new
ArrayList<CORSRule.AllowedMethods>();
        rule2AM.add(CORSRule.AllowedMethods.GET);
        CORSRule rule2 = new
CORSRule().withId("CORSRule2").withAllowedMethods(rule2AM)
                .withAllowedOrigins(Arrays.asList("*")).withMaxAgeSeconds(3000)
                .withExposedHeaders(Arrays.asList("x-amz-server-side-encryption"));

        List<CORSRule> rules = new ArrayList<CORSRule>();
        rules.add(rule1);
        rules.add(rule2);

        // Add the rules to a new CORS configuration.
        BucketCrossOriginConfiguration configuration = new
BucketCrossOriginConfiguration();
        configuration.setRules(rules);

        try {
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                    .withCredentials(new ProfileCredentialsProvider())
                    .withRegion(clientRegion)
                    .build();

            // Add the configuration to the bucket.
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Retrieve and display the configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);

            // Add another new rule.
            List<CORSRule.AllowedMethods> rule3AM = new
ArrayList<CORSRule.AllowedMethods>();
            rule3AM.add(CORSRule.AllowedMethods.HEAD);
            CORSRule rule3 = new
CORSRule().withId("CORSRule3").withAllowedMethods(rule3AM)
                    .withAllowedOrigins(Arrays.asList("http://www.example.com"));

            rules = configuration.getRules();
            rules.add(rule3);
```

```
            configuration.setRules(rules);
            s3Client.setBucketCrossOriginConfiguration(bucketName, configuration);

            // Verify that the new rule was added by checking the number of rules in
the
            // configuration.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            System.out.println("Expected # of rules = 3, found " +
configuration.getRules().size());

            // Delete the configuration.
            s3Client.deleteBucketCrossOriginConfiguration(bucketName);
            System.out.println("Removed CORS configuration.");

            // Retrieve and display the configuration to verify that it was
            // successfully deleted.
            configuration = s3Client.getBucketCrossOriginConfiguration(bucketName);
            printCORSConfiguration(configuration);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }

    private static void printCORSConfiguration(BucketCrossOriginConfiguration
configuration) {
        if (configuration == null) {
            System.out.println("Configuration is null.");
        } else {
            System.out.println("Configuration has " +
configuration.getRules().size() + " rules\n");

            for (CORSRule rule : configuration.getRules()) {
                System.out.println("Rule ID: " + rule.getId());
                System.out.println("MaxAgeSeconds: " + rule.getMaxAgeSeconds());
                System.out.println("AllowedMethod: " + rule.getAllowedMethods());
                System.out.println("AllowedOrigins: " + rule.getAllowedOrigins());
                System.out.println("AllowedHeaders: " + rule.getAllowedHeaders());
                System.out.println("ExposeHeader: " + rule.getExposedHeaders());
```

```
                System.out.println();
            }
        }
    }
}
```

.NET

Example

For information about setting up and running the code examples, see Getting Started with the AWS SDK for .NET in the *AWS SDK for .NET Developer Guide*.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class CORSTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
 RegionEndpoint.USWest2;
        private static IAmazonS3 s3Client;

        public static void Main()
        {
            s3Client = new AmazonS3Client(bucketRegion);
            CORSConfigTestAsync().Wait();
        }
        private static async Task CORSConfigTestAsync()
        {
            try
            {
                // Create a new configuration request and add two rules
                CORSConfiguration configuration = new CORSConfiguration
                {
                    Rules = new System.Collections.Generic.List<CORSRule>
```

```
                        {
                          new CORSRule
                          {
                            Id = "CORSRule1",
                            AllowedMethods = new List<string> {"PUT", "POST",
 "DELETE"},
                            AllowedOrigins = new List<string> {"http://
*.example.com"}
                          },
                          new CORSRule
                          {
                            Id = "CORSRule2",
                            AllowedMethods = new List<string> {"GET"},
                            AllowedOrigins = new List<string> {"*"},
                            MaxAgeSeconds = 3000,
                            ExposeHeaders = new List<string> {"x-amz-server-side-
encryption"}
                          }
                        }
                };

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);

                // Retrieve an existing configuration.
                configuration = await RetrieveCORSConfigurationAsync();

                // Add a new rule.
                configuration.Rules.Add(new CORSRule
                {
                    Id = "CORSRule3",
                    AllowedMethods = new List<string> { "HEAD" },
                    AllowedOrigins = new List<string> { "http://www.example.com" }
                });

                // Add the configuration to the bucket.
                await PutCORSConfigurationAsync(configuration);

                // Verify that there are now three rules.
                configuration = await RetrieveCORSConfigurationAsync();
                Console.WriteLine();
                Console.WriteLine("Expected # of rulest=3; found:{0}",
 configuration.Rules.Count);
                Console.WriteLine();
```

```
                Console.WriteLine("Pause before configuration delete. To continue,
click Enter...");
                Console.ReadKey();

                // Delete the configuration.
                await DeleteCORSConfigurationAsync();

                // Retrieve a nonexistent configuration.
                configuration = await RetrieveCORSConfigurationAsync();
            }
            catch (AmazonS3Exception e)
            {
                Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
            catch (Exception e)
            {
                Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
            }
        }

        static async Task PutCORSConfigurationAsync(CORSConfiguration configuration)
        {

            PutCORSConfigurationRequest request = new PutCORSConfigurationRequest
            {
                BucketName = bucketName,
                Configuration = configuration
            };

            var response = await s3Client.PutCORSConfigurationAsync(request);
        }

        static async Task<CORSConfiguration> RetrieveCORSConfigurationAsync()
        {
            GetCORSConfigurationRequest request = new GetCORSConfigurationRequest
            {
                BucketName = bucketName

            };
            var response = await s3Client.GetCORSConfigurationAsync(request);
            var configuration = response.Configuration;
            PrintCORSRules(configuration);
```

```
            return configuration;
        }

        static async Task DeleteCORSConfigurationAsync()
        {
            DeleteCORSConfigurationRequest request = new
 DeleteCORSConfigurationRequest
            {
                BucketName = bucketName
            };
            await s3Client.DeleteCORSConfigurationAsync(request);
        }

        static void PrintCORSRules(CORSConfiguration configuration)
        {
            Console.WriteLine();

            if (configuration == null)
            {
                Console.WriteLine("\nConfiguration is null");
                return;
            }

            Console.WriteLine("Configuration has {0} rules:",
 configuration.Rules.Count);
            foreach (CORSRule rule in configuration.Rules)
            {
                Console.WriteLine("Rule ID: {0}", rule.Id);
                Console.WriteLine("MaxAgeSeconds: {0}", rule.MaxAgeSeconds);
                Console.WriteLine("AllowedMethod: {0}", string.Join(", ",
 rule.AllowedMethods.ToArray()));
                Console.WriteLine("AllowedOrigins: {0}", string.Join(", ",
 rule.AllowedOrigins.ToArray()));
                Console.WriteLine("AllowedHeaders: {0}", string.Join(", ",
 rule.AllowedHeaders.ToArray()));
                Console.WriteLine("ExposeHeader: {0}", string.Join(", ",
 rule.ExposeHeaders.ToArray()));
            }
        }
    }
}
```

Using the REST API

To set a CORS configuration on your bucket, you can use the AWS Management Console. If your application requires it, you can also send REST requests directly. The following sections in the *Amazon Simple Storage Service API Reference* describe the REST API actions related to the CORS configuration:

- PutBucketCors
- GetBucketCors
- DeleteBucketCors
- OPTIONS object

Testing CORS

To test your CORS configuration, a CORS preflight request can be sent with the `OPTIONS` method so that the server can respond if it is acceptable to send the request. When Amazon S3 receives a preflight request, S3 evaluates the CORS configuration for the bucket and uses the first `CORSRule` rule that matches the incoming request to enable a cross-origin request. For a rule to match, the following conditions must be met:

- The `Origin` header in a CORS request to your bucket must match the origins in the `AllowedOrigins` element in your CORS configuration.
- The HTTP methods that are specified in the `Access-Control-Request-Method` in a CORS request to your bucket must match the method or methods listed in the `AllowedMethods` element in your CORS configuration.
- The headers listed in the `Access-Control-Request-Headers` header in a preflight request must match the headers in the `AllowedHeaders` element in your CORS configuration.

The following is an example of a CORS configuration. To create a CORS Configuration, see Configuring CORS. For more examples of a CORS configuration, see Elements of a CORS configuration.

For guidance on configuring and troubleshooting CORS rules, see How do I configure CORS in Amazon S3 and confirm the CORS rules using cURL? in the AWS re:Post Knowledge Center.

JSON

```
[
    {
        "AllowedHeaders": [
            "Authorization"
        ],
        "AllowedMethods": [
            "GET",
            "PUT",
            "POST",
            "DELETE"
        ],
        "AllowedOrigins": [
            "http://www.example1.com"
        ],
        "ExposeHeaders":  [
             "x-amz-meta-custom-header"
        ]

    }
]
```

To test the CORS configuration, you can send a preflight `OPTIONS` check by using the following CURL command. CURL is a command-line tool that can be used to interact with S3. For more information, see CURL.

```
curl -v -X OPTIONS \
 -H "Origin: http://www.example1.com" \
 -H "Access-Control-Request-Method: PUT" \
 -H "Access-Control-Request-Headers: Authorization" \
 -H "Access-Control-Expose-Headers: x-amz-meta-custom-header"\
    "http://bucket_name.s3.amazonaws.com/object_prefix_name"
```

In the above example, the `curl -v -x OPTIONS` command is used to send a preflight request to S3 to inquire if it is allowed by S3 to send a PUT request on an object from the cross origin `http://www.example1.com`. The headers `Access-Control-Request-Headers` and `Access-Control-Expose-Headers` are optional.

- In response to the `Access-Control-Request-Method` header in the preflight `OPTIONS` request, Amazon S3 returns the list of allowed methods if the requested methods match.
- In response to the `Access-Control-Request-Headers` header in the preflight `OPTIONS` request, Amazon S3 returns the list of allowed headers if the requested headers match.
- In response to the `Access-Control-Expose-Headers` header in the preflight `OPTIONS` request, Amazon S3 returns a list of allowed headers if the requested headers match the allowed headers that can be accessed by scripts running in the browser.

> **Note**
>
> When sending a preflight request, if any of the CORS request headers are not allowed, none of the response CORS headers are returned.

In response to this preflight `OPTIONS` request, you will receive a `200 OK` response. For common error codes received when testing CORS and more information to solve CORS related issues, see Troubleshooting CORS.

```
< HTTP/1.1 200 OK
< Date: Fri, 12 Jul 2024 00:23:51 GMT
< Access-Control-Allow-Origin: http://www.example1.com
< Access-Control-Allow-Methods: GET, PUT, POST, DELETE
< Access-Control-Allow-Headers: Authorization
< Access-Control-Expose-Headers: x-amz-meta-custom-header
< Access-Control-Allow-Credentials: true
< Vary: Origin, Access-Control-Request-Headers, Access-Control-Request-Method
< Server: AmazonS3
< Content-Length: 0
```

Troubleshooting CORS

The following topics can help you troubleshoot some common CORS issues related to S3.

Topics

- 403 Forbidden error: CORS is not enabled for this bucket
- 403 Forbidden error: This CORS request is not allowed
- Headers not found in CORS response

- Considerations of CORS on S3 proxy integrations

403 Forbidden error: CORS is not enabled for this bucket

The following `403 Forbidden` error occurs when a cross-origin request is sent to Amazon S3 but CORS is not configured on your S3 bucket.

Error: HTTP/1.1 403 Forbidden CORS Response: CORS is not enabled for this bucket.

The CORS configuration is a document or policy with rules that identify the origins that you will allow to access your bucket, the operations (HTTP methods) that you will support for each origin, and other operation-specific information. See how to configure CORS on S3 by using the Amazon S3 console, AWS SDKs, and REST API. For more information on CORS and examples of a CORS configuration, see Elements of CORS.

403 Forbidden error: This CORS request is not allowed

The following `403 Forbidden` error is received when a CORS rule in your CORS configuration doesn't match the data in your request.

Error: HTTP/1.1 403 Forbidden CORS Response: This CORS request is not allowed.

As a result, this `403 Forbidden` error can occur for multiple reasons:

- Origin is not allowed.
- Methods are not allowed.
- Requested headers are not allowed.

For each request that Amazon S3 receives, you must have a CORS rule in your CORS configuration that matches the data in your request.

Origin is not allowed

The `Origin` header in a CORS request to your bucket must match the origins in the `AllowedOrigins` element in your CORS configuration. A wildcard character ("`*`") in the `AllowedOrigins` element would match all HTTP methods. For more information on how to update the `AllowedOrigins` element, see Configuring cross-origin resource sharing (CORS).

For example, if only the `http://www.example1.com` domain is included in the `AllowedOrigins` element, then a CORS request sent from the `http://www.example2.com` domain would receive the `403 Forbidden` error.

The following example shows part of a CORS configuration that includes the `http://www.example1.com` domain in the `AllowedOrigins` element.

```
"AllowedOrigins":[
   "http://www.example1.com"
]
```

For a CORS request sent from the `http://www.example2.com` domain to be successful, the `http://www.example2.com` domain should be included in the `AllowedOrigins` element of CORS configuration.

```
"AllowedOrigins":[
   "http://www.example1.com"
   "http://www.example2.com"
]
```

Methods are not allowed

The HTTP methods that are specified in the `Access-Control-Request-Method` in a CORS request to your bucket must match the method or methods listed in the `AllowedMethods` element in your CORS configuration. A wildcard character ("`*`") in `AllowedMethods` would match all HTTP methods. For more information on how to update the `AllowedOrigins` element, see Configuring cross-origin resource sharing (CORS).

In a CORS configuration, you can specify the following methods in the `AllowedMethods` element:

- `GET`
- `PUT`
- `POST`
- `DELETE`
- `HEAD`

The following example shows part of a CORS configuration that includes the `GET` method in the `AllowedMethods` element. Only requests including the `GET` method would succeed.

```
"AllowedMethods":[
   "GET"
]
```

If an HTTP method (for example, PUT) was used in a CORS request or included in a pre-flight CORS request to your bucket but the method isn't present in your CORS configuration, the request would result in a `403 Forbidden` error. To allow this CORS request or CORS pre-flight request, the PUT method must be added to your CORS configuration.

```
"AllowedMethods":[
   "GET"
   "PUT"
]
```

Requested headers are not allowed

The headers listed in the `Access-Control-Request-Headers` header in a pre-flight request must match the headers in the `AllowedHeaders` element in your CORS configuration. For a list of common headers that can be used in requests to Amazon S3, see Common Request Headers. For more information on how to update the `AllowedHeaders` element, see Configuring cross-origin resource sharing (CORS).

The following example shows part of a CORS configuration that includes the `Authorization` header in the `AllowedHeaders` element. Only requests for the `Authorization` header would succeed.

```
"AllowedHeaders":  [
    "Authorization"
]
```

If a header (for example `Content-MD5` was included in a CORS request but the header isn't present in your CORS configuration, the request would result in a `403 Forbidden` error. To allow this CORS request , the `Content-MD5` header must be added to your CORS configuration. If you want to pass both `Authorization` and `Content-MD5` headers in a CORS request to your bucket, confirm that both headers are included in the `AllowedHeaders` element in your CORS configuration.

```
"AllowedHeaders":  [
    "Authorization"
```

```
    "Content-MD5"
]
```

Headers not found in CORS response

The `ExposeHeaders` element in your CORS configuration identifies which response headers that you would like to make accessible to scripts and applications running in browsers, in response to a CORS request.

If your objects stored in your S3 bucket have user-defined metadata (for example, `x-amz-meta-custom-header`) along with the response data, this custom header could contain additional metadata or information that you want to access from your client-side JavaScript code. However, by default, browsers block access to custom headers for security reasons. To allow your client-side JavaScript to access custom headers, you need to include the header in your CORS configuration.

In the example below, the `x-amz-meta-custom-header1` header is included in the `ExposeHeaders` element. The `x-amz-meta-custom-header2` isn't included in the `ExposeHeaders` element and is missing from the CORS configuration. In the response, only the values included in the `ExposeHeaders` element would be returned. If the request included the `x-amz-meta-custom-header2` header in the `Access-Control-Expose-Headers` header, the response would still return a `200 OK`. However, only the permitted header, For example `x-amz-meta-custom-header` would be returned and show in the response.

```
"ExposeHeaders":  [
    "x-amz-meta-custom-header1"
]
```

To ensure all headers appear in the response, add all permitted headers to the `ExposeHeaders` element in your CORS configuration as shown below.

```
"ExposeHeaders":  [
    "x-amz-meta-custom-header1",
    "x-amz-meta-custom-header2"
]
```

Considerations of CORS on S3 proxy integrations

If you are experiencing errors and have already checked the CORS configuration on your S3 bucket, and the cross-origin request is sent to proxies such as AWS CloudFront, try the following:

- Configure the settings to allow the `OPTIONS` method for HTTP requests.
- Configure the proxy to forward the following headers: `Origin`, `Access-Control-Request-Headers`, and `Access-Control-Request-Method`.
- Configure the proxy settings to include the origin header in its cache key. This is important because caching proxies that don't include the origin header in their cache key may serve cached responses that don't include the appropriate CORS headers for different origins.

Some proxies provide pre-defined features for CORS requests. For example, in CloudFront, you can configure a policy that includes the headers

that enable cross-origin resource sharing (CORS) requests when the origin is an Amazon S3 bucket.

This policy has the following settings:

- Headers included in origin requests:

  `Origin`

  `Access-Control-Request-Headers`

  `Access-Control-Request-Method`
- Cookies included in origin requests: None
- Query strings included in origin requests: None

For more information, see Control origin requests with a policy and Use managed origin request policies in the *CloudFront Developer Guide*.

Static website tutorials

The following tutorials or walkthroughs present complete procedures for how to create and configure an Amazon S3 general purpose bucket for static website hosting and hosting on-demand video streaming. The purpose of these tutorials is to provide general guidance. These tutorials are intended for a lab-type environment, and they use example bucket names, user names, and so on. They are not intended for direct use in a production environment without careful review and adaptation to meet the unique needs of your organization's environment.

- Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53 – You can use Amazon S3 with Amazon CloudFront to host videos for on-demand

viewing in a secure and scalable way. After your video is packaged into the right formats, you can store it on a server or in an S3 general purpose bucket, and then deliver it with CloudFront as viewers request it. In this tutorial, you will learn how to configure your general purpose bucket to host on-demand video streaming using CloudFront for delivery and Amazon Route 53 for Domain Name System (DNS) and custom domain management. CloudFront serves the video from its cache, retrieving it from your general purpose bucket only if it is not already cached. This caching management feature accelerates the delivery of your video to viewers globally with low latency, high throughput, and high transfer speeds. For more information about CloudFront caching management, see Optimizing caching and availability in the *Amazon CloudFront Developer Guide*.

- Configuring a static website – You can configure a general purpose bucket to function like a website. This tutorial walks you through the steps of hosting a website on Amazon S3 including creating a bucket, enabling static website hosting in the S3 console, creating an index document and creating an error document. For more information, see Hosting a static website using Amazon S3.
- Configuring a static website using a custom domain registered with Route 53 – You can create and configure a general purpose bucket to host a static website and create redirects on S3 for a website with a custom domain name that is registered with Amazon Route 53. You use Route 53 to register domains and to define where you want to route internet traffic for your domain. This tutorial shows how to create Route 53 alias records that routes traffic for your domain and subdomain to your general purpose bucket that contains an HTML file. For more information, see Use your domain for a static website in an Amazon S3 bucket in the *Amazon Route 53 Developer Guide*. After you complete this tutorial, you can optionally use CloudFront to improve the performance of your website. For more information, see Speeding up your website with Amazon CloudFront.
- Deploying a static website to AWS Amplify Hosting from an S3 general purpose bucket – We recommend that you use AWS Amplify Hosting to host static website content stored on S3. Amplify Hosting is a fully managed service that makes it easy to deploy your websites on a globally available content delivery network (CDN) powered by Amazon CloudFront, allowing secure static website hosting without extensive setup. With AWS Amplify Hosting, you can select the location of your objects within your general purpose bucket, deploy your content to a managed CDN, and generate a public HTTPS URL for your website to be accessible anywhere. For more information, see Deploying a static website from S3 using the Amplify console in the *AWS Amplify Hosting User Guide*.

Tutorial: Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53

You can use Amazon S3 with Amazon CloudFront to host videos for on-demand viewing in a secure and scalable way. Video on demand (VOD) streaming means that your video content is stored on a server and viewers can watch it at any time.

CloudFront is a fast, highly secure, and programmable content delivery network (CDN) service. CloudFront can deliver your content securely over HTTPS from all of the CloudFront edge locations around the globe. For more information about CloudFront, see What is Amazon CloudFront? in the *Amazon CloudFront Developer Guide*.

CloudFront caching reduces the number of requests that your origin server must respond to directly. When a viewer (end user) requests a video that you serve with CloudFront, the request is routed to a nearby edge location closer to where the viewer is located. CloudFront serves the video from its cache, retrieving it from the S3 bucket only if it is not already cached. This caching management feature accelerates the delivery of your video to viewers globally with low latency, high throughput, and high transfer speeds. For more information about CloudFront caching management, see Optimizing caching and availability in the *Amazon CloudFront Developer Guide*.



Objective

In this tutorial, you configure an S3 bucket to host on-demand video streaming using CloudFront for delivery and Amazon Route 53 for Domain Name System (DNS) and custom domain management.

Topics

- Prerequisites: Register and configure a custom domain with Route 53
- Step 1: Create an S3 bucket
- Step 2: Upload a video to the S3 bucket
- Step 3: Create a CloudFront origin access identity
- Step 4: Create a CloudFront distribution
- Step 5: Access the video through the CloudFront distribution
- Step 6: Configure your CloudFront distribution to use your custom domain name
- Step 7: Access the S3 video through the CloudFront distribution with the custom domain name
- (Optional) Step 8: View data about requests received by your CloudFront distribution
- Step 9: Clean up
- Next steps

Prerequisites: Register and configure a custom domain with Route 53

Before you start this tutorial, you must register and configure a custom domain (for example, **`example.com`**) with Route 53 so that you can configure your CloudFront distribution to use a custom domain name later.

Without a custom domain name, your S3 video is publicly accessible and hosted through CloudFront at a URL that looks similar to the following:

```
https://CloudFront distribution domain name/Path to an S3 video
```

For example, **`https://d111111abcdef8.cloudfront.net/sample.mp4`**.

After you configure your CloudFront distribution to use a custom domain name configured with Route 53, your S3 video is publicly accessible and hosted through CloudFront at a URL that looks similar to the following:

```
https://CloudFront distribution alternate domain name/Path to an S3 video
```

For example, **`https://www.example.com/sample.mp4`**. A custom domain name is simpler and more intuitive for your viewers to use.

To register a custom domain, see Registering a new domain using Route 53 in the *Amazon Route 53 Developer Guide*.

When you register a domain name with Route 53, Route 53 creates the hosted zone for you, which you will use later in this tutorial. This hosted zone is where you store information about how to route traffic for your domain, for example, to an Amazon EC2 instance or a CloudFront distribution.

There are fees associated with domain registration, your hosted zone, and DNS queries received by your domain. For more information, see Amazon Route 53 Pricing.

> **Note**
>
> When you register a domain, it costs money immediately and it's irreversible. You can choose not to auto-renew the domain, but you pay up front and own it for the year. For more information, see Registering a new domain in the *Amazon Route 53 Developer Guide*.

Step 1: Create an S3 bucket

Create a bucket to store the original video that you plan to stream.

To create a bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

   > **Note**
   >
   > To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.

3. In the left navigation pane, choose **General purpose buckets**.
4. Choose **Create bucket**. The **Create bucket** page opens.
5. For **Bucket name**, enter a name for your bucket (for example, **`tutorial-bucket`**).

   For more information about naming buckets in Amazon S3, see General purpose bucket naming rules.
6. For **Region**, choose the AWS Region where you want the bucket to reside.

   If possible, you should pick the Region that is closest to the majority of your viewers. For more information about the bucket Region, see General purpose buckets overview.
7. For **Block Public Access settings for this bucket**, keep the default settings (**Block *all* public access** is enabled).

   Even with **Block *all* public access** enabled, viewers can still access the uploaded video through CloudFront. This feature is a major advantage of using CloudFront to host a video stored in S3.

   We recommend that you keep all settings enabled unless you need to turn off one or more of them for your use case. For more information about blocking public access, see Blocking public access to your Amazon S3 storage.
8. For the remaining settings, keep the defaults.

   (Optional) If you want to configure additional bucket settings for your specific use case, see Creating a general purpose bucket.
9. Choose **Create bucket**.

Step 2: Upload a video to the S3 bucket

The following procedure describes how to upload a video file to an S3 bucket by using the console. If you're uploading many large video files to S3, you might want to use Amazon S3 Transfer Acceleration to configure fast and secure file transfers. Transfer Acceleration can speed up video uploading to your S3 bucket for long-distance transfer of larger videos. For more information, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

To upload a file to the bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the left navigation pane, choose **General purpose buckets**.
3. In the **General purpose buckets** list, choose the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**) to upload your file to.
4. On the **Objects** tab for your bucket, choose **Upload**.
5. On the **Upload** page, under **Files and folders**, choose **Add files**.
6. Choose a file to upload, and then choose **Open**.

   For example, you can upload a video file named `sample.mp4`.
7. Choose **Upload**.

Step 3: Create a CloudFront origin access identity

To restrict direct access to the video from your S3 bucket, create a special CloudFront user called an origin access identity (OAI). You will associate the OAI with your distribution later in this tutorial. By using an OAI, you make sure that viewers can't bypass CloudFront and get the video directly from the S3 bucket. Only the CloudFront OAI can access the file in the S3 bucket. For more information, see Restrict access to an Amazon S3 origin in the *Amazon CloudFront Developer Guide*.

> **Important**
>
> If the bucket that you're using to host your static website has been encrypted using server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS), you must use origin access control (OAC) instead of origin access identity (OAI) to secure the origin. OAI doesn't support SSE-KMS, so you must use OAC instead. For more information about OAC, see Restrict access to an Amazon S3 origin in the *Amazon CloudFront Developer Guide*.

To create a CloudFront OAI

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, under the **Security** section, choose **Origin access**.
3. Under the **Identities** tab, choose **Create origin access identity**.
4. Enter a name (for example, **`S3-OAI`**) for the new origin access identity.
5. Choose **Create**.

Step 4: Create a CloudFront distribution

To use CloudFront to serve and distribute the video in your S3 bucket, you must create a CloudFront distribution.

Substeps

- Create a CloudFront distribution
- Review the bucket policy

Create a CloudFront distribution

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, choose **Distributions**.
3. Choose **Create distribution**.
4. In the **Origin** section, for **Origin domain**, choose the domain name of your S3 origin, which starts with the name of the S3 bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
5. For **Origin access**, choose **Legacy access identities**.
6. Under **Origin access identity**, choose the origin access identity that you created in Step 3 (for example, **`S3-OAI`**).
7. Under **Bucket policy**, choose **Yes, update the bucket policy**.
8. In the **Default cache behavior** section, under **Viewer protocol policy**, choose **Redirect HTTP to HTTPS**.

   When you choose this feature, HTTP requests are automatically redirected to HTTPS to secure your website and protect your viewers' data.
9. For the other settings in the **Default cache behaviors** section, keep the default values.

   (Optional) You can control how long your file stays in a CloudFront cache before CloudFront forwards another request to your origin. Reducing the duration allows you to serve dynamic content. Increasing the duration means that your viewers get better performance because your files are more likely to be served directly from the edge cache. A longer duration also reduces the load on your origin. For more information, see Managing how long content stays in the cache (expiration) in the *Amazon CloudFront Developer Guide*.
10. For the other sections, keep the remaining settings set to the defaults.

For more information about the different settings options, see Values That You Specify When You Create or Update a Distribution in the *Amazon CloudFront Developer Guide*.

11. Choose **Create distribution**.
12. On the **General** tab for your CloudFront distribution, under **Details**, the value of the **Last modified** column for your distribution changes from **Deploying** to the timestamp when the distribution was last modified. This process typically takes a few minutes.

Review the bucket policy

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the name of the bucket that you used earlier as the origin of your CloudFront distribution (for example, **`tutorial-bucket`**).
4. Choose the **Permissions** tab.
5. In the **Bucket policy** section, confirm that you see a statement similar to the following in the bucket policy text:

```
{
    "Version": "2008-10-17",
    "Id": "PolicyForCloudFrontPrivateContent",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::cloudfront:user/CloudFront Origin Access
 Identity EH1HDMB1FH2TC"
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::tutorial-bucket/*"
        }
    ]
}
```

This is the statement that your CloudFront distribution added to your bucket policy when you chose **Yes, update the bucket policy** earlier.

This bucket policy update indicates that you successfully configured the CloudFront distribution to restrict access to the S3 bucket. Because of this restriction, objects in the bucket can be accessed only through your CloudFront distribution.

Step 5: Access the video through the CloudFront distribution

Now, CloudFront can serve the video stored in your S3 bucket. To access your video through CloudFront, you must combine your CloudFront distribution domain name with the path to the video in the S3 bucket.

To create a URL to the S3 video using the CloudFront distribution domain name

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, choose **Distributions**.
3. To get the distribution domain name, do the following:
   a. In the **Origins** column, find the correct CloudFront distribution by looking for its origin name, which starts with the S3 bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
   b. After finding the distribution in the list, widen the **Domain name** column to copy the domain name value for your CloudFront distribution.
4. In a new browser tab, paste the distribution domain name that you copied.
5. Return to the previous browser tab, and open the S3 console at https://console.aws.amazon.com/s3/.
6. In the left navigation pane, choose **Buckets**.
7. In the **Buckets** list, choose the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
8. In the **Objects** list, choose the name of the video that you uploaded in Step 2 (for example, `sample.mp4`).
9. On the object detail page, in the **Object overview** section, copy the value of the **Key**. This value is the path to the uploaded video object in the S3 bucket.
10. Return to the browser tab where you previously pasted the distribution domain name, enter a forward slash (**/**) after the distribution domain name, and then paste the path to the video that you copied earlier (for example, `sample.mp4`).

Now, your S3 video is publicly accessible and hosted through CloudFront at a URL that looks similar to the following:

```
https://CloudFront distribution domain name/Path to the S3 video
```

Replace *CloudFront distribution domain name* and *Path to the S3 video* with the appropriate values. An example URL is **https://d111111abcdef8.cloudfront.net/sample.mp4**.

Step 6: Configure your CloudFront distribution to use your custom domain name

To use your own domain name instead of the CloudFront domain name in the URL to access the S3 video, add an alternate domain name to your CloudFront distribution.

Substeps

- Request an SSL certificate
- Add the alternate domain name to your CloudFront distribution
- Create a DNS record to route traffic from your alternate domain name to your CloudFront distribution's domain name
- Check whether IPv6 is enabled for your distribution and create another DNS record if needed

Request an SSL certificate

To allow your viewers to use HTTPS and your custom domain name in the URL for your video streaming, use AWS Certificate Manager (ACM) to request a Secure Sockets Layer (SSL) certificate. The SSL certificate establishes an encrypted network connection to the website.

1. Sign in to the AWS Management Console and open the ACM console at https://console.aws.amazon.com/acm/.
2. If the introductory page appears, under **Provision certificates**, choose **Get Started**.
3. On the **Request a certificate** page, choose **Request a public certificate**, and then choose **Request a certificate**.
4. On the **Add domain names** page, enter the fully qualified domain name (FQDN) of the site that you want to secure with an SSL/TLS certificate. You can use an asterisk (*) to request a wildcard certificate to protect several site names in the same domain. For this tutorial, enter

***** and the custom domain name that you configured in Prerequisites. For example, enter **`*.example.com`**, and then choose **Next**.

For more information, see To request an ACM public certificate (console) in the *AWS Certificate Manager User Guide*.

5. On the **Select validation method** page, choose **DNS validation**. Then, choose **Next**.

   If you are able to edit your DNS configuration, we recommend that you use DNS domain validation rather than email validation. DNS validation has multiple benefits over email validation. For more information, see Option 1: DNS validation in the *AWS Certificate Manager User Guide*.
6. (Optional) On the **Add tags** page, tag your certificate with metadata.
7. Choose **Review**.
8. On the **Review** page, verify that the information under **Domain name** and **Validation method** are correct. Then, choose **Confirm and request**.

   The **Validation** page shows that your request is being processed and that the certificate domain is being validated. The certificate awaiting validation is in the **Pending validation** status.
9. On the **Validation** page, choose the down arrow to the left of your custom domain name, and then choose **Create record in Route 53** to validate your domain ownership through DNS.

   Doing this adds a CNAME record provided by AWS Certificate Manager to your DNS configuration.
10. In the **Create record in Route 53** dialog box, choose **Create**.

    The **Validation** page should display a **Success** status notification.
11. Choose **Continue** to view the **Certificates** list page.

    The **Status** for your new certificate changes from **Pending validation** to **Issued** within 30 minutes.

Add the alternate domain name to your CloudFront distribution

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, choose **Distributions**.

3. Choose the ID for the distribution that you created in Step 4.
4. On the **General** tab, go to the **Settings** section, and choose **Edit**.
5. On the **Edit settings** page, for **Alternate domain name (CNAME) - *optional***, choose **Add item** to add the custom domain names that you want to use in the URL for the S3 video served by this CloudFront distribution.

   In this tutorial, for example, if you want to route traffic for a subdomain, such as `www.example.com`, enter the subdomain name (`www`) with the domain name (`example.com`). Specifically, enter **`www.example.com`**.

   > **Note**
   >
   > The alternate domain name (CNAME) that you add must be covered by the SSL certificate that you previously attached to your CloudFront distribution.

6. For **Custom SSL certificate - *optional***, choose the SSL certificate that you requested earlier (for example, **`*.example.com`**).

   > **Note**
   >
   > If you don't see the SSL certificate immediately after you request it, wait 30 minutes, and then refresh the list until the SSL certificate is available for you to select.

7. Keep the remaining settings set to the defaults. Choose **Save changes**.
8. On the **General** tab for the distribution, wait for the value of **Last modified** to change from **Deploying** to the timestamp when the distribution was last modified.

Create a DNS record to route traffic from your alternate domain name to your CloudFront distribution's domain name

1. Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.
2. In the left navigation pane, choose **Hosted zones**.
3. On the **Hosted zones** page, choose the name of the hosted zone that Route 53 created for you in Prerequisites (for example, **`example.com`**).
4. Choose **Create record**, and then use the **Quick create record** method.

5. For **Record name**, keep the value for the record name the same as the alternate domain name of the CloudFront distribution that you added earlier.

   In this tutorial, to route traffic to a subdomain, such as `www.example.com`, enter the subdomain name without the domain name. For example, enter only **`www`** in the text field before your custom domain name.
6. For **Record type**, choose **A - Routes traffic to an IPv4 address and some AWS resources**.
7. For **Value**, choose the **Alias** toggle to enable the alias resource.
8. Under **Route traffic to**, choose **Alias to CloudFront distribution** from the dropdown list.
9. In the search box that says **Choose distribution**, choose the domain name of the CloudFront distribution that you created in Step 4.

   To find the domain name of your CloudFront distribution, do the following:

   a. In a new browser tab, sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v3/home.
   b. In the left navigation pane, choose **Distributions**.
   c. In the **Origins** column, find the correct CloudFront distribution by looking for its origin name, which starts with the S3 bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
   d. After finding the distribution in the list, widen the **Domain name** column to see the domain name value for your CloudFront distribution.
10. On the **Create record** page in the Route 53 console, for the remaining settings, keep the defaults.
11. Choose **Create records**.

Check whether IPv6 is enabled for your distribution and create another DNS record if needed

If IPv6 is enabled for your distribution, you must create another DNS record.

1. To check whether IPv6 is enabled for your distribution, do the following:

   a. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
   b. In the left navigation pane, choose **Distributions**.
   c. Choose the ID of the CloudFront distribution that you created in Step 4.

d. On the **General** tab, under **Settings**, check whether **IPv6** is set to **Enabled**.

If IPv6 is enabled for your distribution, you must create another DNS record.

2. If IPv6 is enabled for your distribution, do the following to create a DNS record:

a. Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.

b. In the left navigation pane, choose **Hosted zones**.

c. On the **Hosted zones** page, choose the name of the hosted zone that Route 53 created for you in Prerequisites (for example, **`example.com`**).

d. Choose **Create record**, and then use the **Quick create record** method.

e. For **Record name**, in the text field before your custom domain name, type the same value that you typed when you created the IPv4 DNS record earlier. For example, in this tutorial, to route traffic for the subdomain `www.example.com`, enter only **`www`**.

f. For **Record type**, choose **AAAA - Routes traffic to an IPv6 address and some AWS resources**.

g. For **Value**, choose the **Alias** toggle to enable the alias resource.

h. Under **Route traffic to**, choose **Alias to CloudFront distribution** from the dropdown list.

i. In the search box that says **Choose distribution**, choose the domain name of the CloudFront distribution that you created in Step 4.

j. For the remaining settings, keep the defaults.

k. Choose **Create records**.

Step 7: Access the S3 video through the CloudFront distribution with the custom domain name

To access the S3 video using the custom URL, you must combine your alternate domain name with the path to the video in the S3 bucket.

To create a custom URL to access the S3 video through the CloudFront distribution

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, choose **Distributions**.
3. To get the alternate domain name of your CloudFront distribution, do the following:

a. In the **Origins** column, find the correct CloudFront distribution by looking for its origin name, which starts with the S3 bucket name for the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).

b. After finding the distribution in the list, widen the **Alternate domain names** column to copy the value of the alternate domain name of your CloudFront distribution.

4. In a new browser tab, paste the alternate domain name of the CloudFront distribution.
5. Return to the previous browser tab, and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
6. Find the path to your S3 video, as explained in Step 5.
7. Return to the browser tab where you previously pasted the alternate domain name, enter a forward slash (**/**), and then paste the path to your S3 video (for example, `sample.mp4`).

   Now, your S3 video is publicly accessible and hosted through CloudFront at a custom URL that looks similar to the following:

   ```
   https://CloudFront distribution alternate domain name/Path to the S3 video
   ```

   Replace *`CloudFront distribution alternate domain name`* and *`Path to the S3 video`* with the appropriate values. An example URL is **`https://www.example.com/sample.mp4`**.

(Optional) Step 8: View data about requests received by your CloudFront distribution

To view data about requests received by your CloudFront distribution

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, under **Reports & analytics**, choose the reports from the console, ranging from **Cache statistics**, **Popular Objects**, **Top Referrers**, **Usage**, and **Viewers**.

   You can filter each report dashboard. For more information, see CloudFront Reports in the Console in the *Amazon CloudFront Developer Guide*.
3. To filter data, choose the ID of the CloudFront distribution that you created in Step 4.

Step 9: Clean up

If you hosted an S3 streaming video using CloudFront and Route 53 only as a learning exercise, delete the AWS resources that you allocated so that you no longer accrue charges.

> **Note**
>
> When you register a domain, it costs money immediately and it's irreversible. You can choose not to auto-renew the domain, but you pay up front and own it for the year. For more information, see Registering a new domain in the *Amazon Route 53 Developer Guide*.

Substeps

- Delete the CloudFront distribution
- Delete the DNS record
- Delete the public hosted zone for your custom domain
- Delete the custom domain name from Route 53
- Delete the original video in the S3 source bucket
- Delete the S3 source bucket

Delete the CloudFront distribution

1. Sign in to the AWS Management Console and open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. In the left navigation pane, choose **Distributions**.
3. In the **Origins** column, find the correct CloudFront distribution by looking for its origin name, which starts with the S3 bucket name for the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
4. To delete the CloudFront distribution, you must disable it first.

   - If the value of the **Status** column is **Enabled** and the value of **Last modified** is the timestamp when the distribution was last modified, continue to disable the distribution before deleting it.
   - If the value of **Status** is **Enabled** and the value of **Last modified** is **Deploying**, wait until the value of **Status** changes to the timestamp when the distribution was last modified. Then continue to disable the distribution before deleting it.

5. To disable the CloudFront distribution, do the following:

   a. In the **Distributions** list, select the check box next to the ID for the distribution that you want to delete.

   b. To disable the distribution, choose **Disable**, and then choose **Disable** to confirm.

      If you disable a distribution that has an alternate domain name associated with it, CloudFront stops accepting traffic for that domain name (such as `www.example.com`), even if another distribution has an alternate domain name with a wildcard (`*`) that matches the same domain (such as `*.example.com`).

   c. The value of **Status** immediately changes to **Disabled**. Wait until the value of **Last modified** changes from **Deploying** to the timestamp when the distribution was last modified.

      Because CloudFront must propagate this change to all edge locations, it might take a few minutes before the update is complete and the **Delete** option is available for you to delete the distribution.

6. To delete the disabled distribution, do the following:

   a. Choose the check box next to the ID for the distribution that you want to delete.

   b. Choose **Delete**, and then choose **Delete** to confirm.

Delete the DNS record

If you want to delete the public hosted zone for the domain (including the DNS record), see Delete the public hosted zone for your custom domain in the *Amazon Route 53 Developer Guide*. If you only want to delete the DNS record created in Step 6, do the following:

1. Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.
2. In the left navigation pane, choose **Hosted zones**.
3. On the **Hosted zones** page, choose the name of the hosted zone that Route 53 created for you in Prerequisites (for example, **`example.com`**).
4. In the list of records, select the check box next to the records that you want to delete (the records that you created in Step 6).

> **Note**
>
> You can't delete records that have a **Type** value of **NS** or **SOA**.

5. Choose **Delete records**.
6. To confirm the deletion, choose **Delete**.

   Changes to records take time to propagate to the Route 53 DNS servers. Currently, the only way to verify that your changes have propagated is to use the GetChange API action. Changes usually propagate to all Route 53 name servers within 60 seconds.

Delete the public hosted zone for your custom domain

> **Warning**
>
> If you want to keep your domain registration but stop routing internet traffic to your website or web application, we recommend that you delete records in the hosted zone (as described in the prior section) instead of deleting the hosted zone.
> If you delete a hosted zone, someone else can use the domain and route traffic to their own resources using your domain name.
> In addition, if you delete a hosted zone, you can't undelete it. You must create a new hosted zone and update the name servers for your domain registration, which can take up to 48 hours to take effect.
> If you want to make the domain unavailable on the internet, you can first transfer your DNS service to a free DNS service and then delete the Route 53 hosted zone. This prevents future DNS queries from possibly being misrouted.
>
> 1. If the domain is registered with Route 53, see Adding or changing name servers and glue records for a domain in the *Amazon Route 53 Developer Guide* for information about how to replace Route 53 name servers with name servers for the new DNS service.
> 2. If the domain is registered with another registrar, use the method provided by the registrar to change name servers for the domain.

> **Note**
>
> If you're deleting a hosted zone for a subdomain (`www.example.com`), you don't need to change name servers for the domain (`example.com`).

1. Sign in to the AWS Management Console and open the Route 53 console at https://console.aws.amazon.com/route53/.
2. In the left navigation pane, choose **Hosted zones**.
3. On the **Hosted zones** page, choose the name of the hosted zone that you want to delete.
4. On the **Records** tab for your hosted zone, confirm that the hosted zone that you want to delete contains only an **NS** and an **SOA** record.

   If it contains additional records, delete them first.

   If you created any NS records for subdomains in the hosted zone, delete those records too.
5. On the **DNSSEC signing** tab for your hosted zone, disable DNNSSEC signing if it was enabled. For more information, see Disabling DNSSEC signing in the *Amazon Route 53 Developer Guide*.
6. On the details page of the hosted zone, choose **Delete zone**.
7. To confirm the deletion, enter **`delete`**, and then choose **Delete**.

Delete the custom domain name from Route 53

For most top-level domains (TLDs), you can delete the registration if you no longer want it. If you delete a domain name registration from Route 53 before the registration is scheduled to expire, AWS does not refund the registration fee. For more information, see Deleting a domain name registration in the *Amazon Route 53 Developer Guide*.

> **Important**
>
> If you want to transfer the domain between AWS accounts or transfer the domain to another registrar, don't delete the domain and expect to immediately reregister it. Instead, see the applicable documentation in the *Amazon Route 53 Developer Guide*:
>
> - Transferring a domain to a different AWS account

- Transferring a domain from Amazon Route 53 to another registrar

Delete the original video in the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Bucket name** list, choose the name of the bucket that you uploaded the video to in Step 2 (for example, **`tutorial-bucket`**).
4. On the **Objects** tab, select the check box next to the name of the object that you want to delete (for example, `sample.mp4`).
5. Choose **Delete**.
6. Under **Permanently delete objects?**, enter **`permanently delete`** to confirm that you want to delete this object.
7. Choose **Delete objects**.

Delete the S3 source bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, select the option button next to the name of the bucket that you created in Step 1 (for example, **`tutorial-bucket`**).
4. Choose **Delete**.
5. On the **Delete bucket** page, confirm that you want to delete the bucket by entering the bucket name in the text field, and then choose **Delete bucket**.

Next steps

After you complete this tutorial, you can further explore the following related use cases:

- Transcode S3 videos into streaming formats needed by a particular television or connected device before hosting these videos with a CloudFront distribution.

To use Amazon S3 Batch Operations, AWS Lambda and AWS Elemental MediaConvert to batch-transcode a collection of videos to a variety of output media formats, see Tutorial: Batch-transcoding videos with S3 Batch Operations.

- Host other objects stored in S3, such as images, audio, motion graphics, style sheets, HTML, JavaScript, React apps, and so on, using CloudFront and Route 53.

  For example, see Tutorial: Configuring a static website using a custom domain registered with Route 53 and Speeding up your website with Amazon CloudFront.
- Use Amazon S3 Transfer Acceleration to configure fast and secure file transfers. Transfer Acceleration can speed up video uploading to your S3 bucket for long-distance transfer of larger videos. Transfer Acceleration improves transfer performance by routing traffic through the CloudFront globally distributed edge locations and over the AWS backbone networks. It also uses network protocol optimizations. For more information, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.

Tutorial: Configuring a static website on Amazon S3

Important

Amazon S3 now applies server-side encryption with Amazon S3 managed keys (SSE-S3) as the base level of encryption for every bucket in Amazon S3. Starting January 5, 2023, all new object uploads to Amazon S3 are automatically encrypted at no additional cost and with no impact on performance. The automatic encryption status for S3 bucket default encryption configuration and for new object uploads is available in CloudTrail logs, S3 Inventory, S3 Storage Lens, the Amazon S3 console, and as an additional Amazon S3 API response header in the AWS CLI and AWS SDKs. For more information, see Default encryption FAQ.

You can configure an Amazon S3 bucket to function like a website. This example walks you through the steps of hosting a website on Amazon S3.

Important

The following tutorial requires disabling Block Public Access. We recommend keeping Block Public Access enabled. If you want to keep all four Block Public Access settings enabled

and host a static website, you can use Amazon CloudFront origin access control (OAC). Amazon CloudFront provides the capabilities required to set up a secure static website. Amazon S3 static websites support only HTTP endpoints. Amazon CloudFront uses the durable storage of Amazon S3 while providing additional security headers, such as HTTPS. HTTPS adds security by encrypting a normal HTTP request and protecting against common cyberattacks. For more information, see Getting started with a secure static website in the *Amazon CloudFront Developer Guide*.

Topics

- Step 1: Create a bucket
- Step 2: Enable static website hosting
- Step 3: Edit Block Public Access settings
- Step 4: Add a bucket policy that makes your bucket content publicly available
- Step 5: Configure an index document
- Step 6: Configure an error document
- Step 7: Test your website endpoint
- Step 8: Clean up

Step 1: Create a bucket

The following instructions provide an overview of how to create your buckets for website hosting. For detailed, step-by-step instructions on creating a bucket, see Creating a general purpose bucket.

To create a bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **Create bucket**.
3. Enter the **Bucket name** (for example, **`example.com`**).
4. Choose the Region where you want to create the bucket.

   Choose a Region that is geographically close to you to minimize latency and costs, or to address regulatory requirements. The Region that you choose determines your Amazon S3 website endpoint. For more information, see Website endpoints.

5. To accept the default settings and create the bucket, choose **Create**.

Step 2: Enable static website hosting

After you create a bucket, you can enable static website hosting for your bucket. You can create a new bucket or use an existing bucket.

To enable static website hosting

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to enable static website hosting for.
4. Choose **Properties**.
5. Under **Static website hosting**, choose **Edit**.
6. Choose **Use this bucket to host a website**.
7. Under **Static website hosting**, choose **Enable**.
8. In **Index document**, enter the file name of the index document, typically `index.html`.

   The index document name is case sensitive and must exactly match the file name of the HTML index document that you plan to upload to your S3 bucket. When you configure a bucket for website hosting, you must specify an index document. Amazon S3 returns this index document when requests are made to the root domain or any of the subfolders. For more information, see Configuring an index document.
9. To provide your own custom error document for 4XX class errors, in **Error document**, enter the custom error document file name.

   The error document name is case sensitive and must exactly match the file name of the HTML error document that you plan to upload to your S3 bucket. If you don't specify a custom error document and an error occurs, Amazon S3 returns a default HTML error document. For more information, see Configuring a custom error document.
10. (Optional) If you want to specify advanced redirection rules, in **Redirection rules**, enter JSON to describe the rules.

For example, you can conditionally route requests according to specific object key names or prefixes in the request. For more information, see Configure redirection rules to use advanced conditional redirects.

11. Choose **Save changes**.

    Amazon S3 enables static website hosting for your bucket. At the bottom of the page, under **Static website hosting**, you see the website endpoint for your bucket.

12. Under **Static website hosting**, note the **Endpoint**.

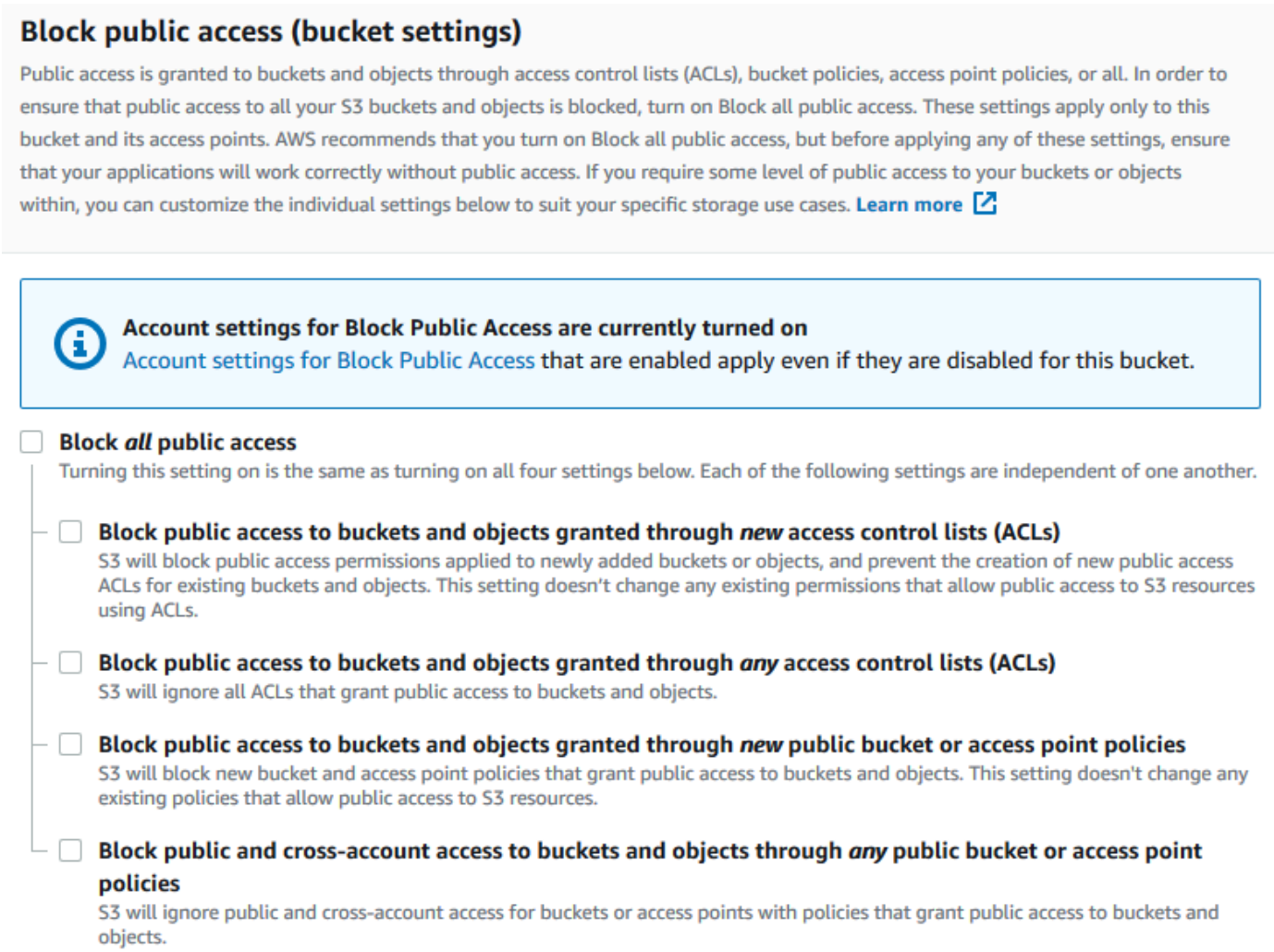
    The **Endpoint** is the Amazon S3 website endpoint for your bucket. After you finish configuring your bucket as a static website, you can use this endpoint to test your website.

Step 3: Edit Block Public Access settings

By default, Amazon S3 blocks public access to your account and buckets. If you want to use a bucket to host a static website, you can use these steps to edit your block public access settings.

> **Warning**
>
> Before you complete these steps, review Blocking public access to your Amazon S3 storage to ensure that you understand and accept the risks involved with allowing public access. When you turn off block public access settings to make your bucket public, anyone on the internet can access your bucket. We recommend that you block all public access to your buckets.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the bucket that you have configured as a static website.
3. Choose **Permissions**.
4. Under **Block public access (bucket settings)**, choose **Edit**.
5. Clear **Block *all* public access**, and choose **Save changes**.



Amazon S3 turns off the Block Public Access settings for your bucket. To create a public static website, you might also have to edit the Block Public Access settings for your account before adding a bucket policy. If the Block Public Access settings for your account are currently turned on, you see a note under **Block public access (bucket settings)**.

Step 4: Add a bucket policy that makes your bucket content publicly available

After you edit S3 Block Public Access settings, you can add a bucket policy to grant public read access to your bucket. When you grant public read access, anyone on the internet can access your bucket.

Important

The following policy is an example only and allows full access to the contents of your bucket. Before you proceed with this step, review How can I secure the files in my Amazon S3 bucket? to ensure that you understand the best practices for securing the files in your S3 bucket and risks involved in granting public access.

1. Under **Buckets**, choose the name of your bucket.
2. Choose **Permissions**.
3. Under **Bucket Policy**, choose **Edit**.
4. To grant public read access for your website, copy the following bucket policy, and paste it in the **Bucket policy editor**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. Update the `Resource` to your bucket name.

   In the preceding example bucket policy, *`Bucket-Name`* is a placeholder for the bucket name. To use this bucket policy with your own bucket, you must update this name to match your bucket name.

6. Choose **Save changes**.

   A message appears indicating that the bucket policy has been successfully added.

If you see an error that says `Policy has invalid resource`, confirm that the bucket name in the bucket policy matches your bucket name. For information about adding a bucket policy, see How do I add an S3 bucket policy?

If you get an error message and cannot save the bucket policy, check your account and bucket Block Public Access settings to confirm that you allow public access to the bucket.

Step 5: Configure an index document

When you enable static website hosting for your bucket, you enter the name of the index document (for example, **`index.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this index document name to your bucket.

To configure the index document

1. Create an `index.html` file.

   If you don't have an `index.html` file, you can use the following HTML to create one:

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" >
   <head>
       <title>My Website Home Page</title>
   </head>
   <body>
     <h1>Welcome to my website</h1>
     <p>Now hosted on Amazon S3!</p>
   </body>
   </html>
   ```

2. Save the index file locally.

   The index document file name must exactly match the index document name that you enter in the **Static website hosting** dialog box. The index document name is case sensitive. For example, if you enter `index.html` for the **Index document** name in the **Static website hosting** dialog box, your index document file name must also be `index.html` and not `Index.html`.

3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

4. In the left navigation pane, choose **General purpose buckets**.

5. In the buckets list, choose the name of the bucket that you want to use to host a static website.
6. Enable static website hosting for your bucket, and enter the exact name of your index document (for example, `index.html`). For more information, see Enabling website hosting.

   After enabling static website hosting, proceed to step 6.
7. To upload the index document to your bucket, do one of the following:

   - Drag and drop the index file into the console bucket listing.
   - Choose **Upload**, and follow the prompts to choose and upload the index file.

   For step-by-step instructions, see Uploading objects.
8. (Optional) Upload other website content to your bucket.

Step 6: Configure an error document

When you enable static website hosting for your bucket, you enter the name of the error document (for example, **`404.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this error document name to your bucket.

To configure an error document

1. Create an error document, for example `404.html`.
2. Save the error document file locally.

   The error document name is case sensitive and must exactly match the name that you enter when you enable static website hosting. For example, if you enter `404.html` for the **Error document** name in the **Static website hosting** dialog box, your error document file name must also be `404.html`.
3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
4. In the left navigation pane, choose **General purpose buckets**.
5. In the buckets list, choose the name of the bucket that you want to use to host a static website.

6. Enable static website hosting for your bucket, and enter the exact name of your error document (for example, `404.html`). For more information, see Enabling website hosting and Configuring a custom error document.

   After enabling static website hosting, proceed to step 6.

7. To upload the error document to your bucket, do one of the following:

   - Drag and drop the error document file into the console bucket listing.
   - Choose **Upload**, and follow the prompts to choose and upload the index file.

   For step-by-step instructions, see Uploading objects.

Step 7: Test your website endpoint

After you configure static website hosting for your bucket, you can test your website endpoint.

> **Note**
>
> Amazon S3 does not support HTTPS access to the website. If you want to use HTTPS, you can use Amazon CloudFront to serve a static website hosted on Amazon S3.
> For more information, see How do I use CloudFront to serve a static website hosted on Amazon S3? and Requiring HTTPS for communication between viewers and CloudFront.

1. Under **Buckets**, choose the name of your bucket.
2. Choose **Properties**.
3. At the bottom of the page, under **Static website hosting**, choose your **Bucket website endpoint**.

   Your index document opens in a separate browser window.

You now have a website hosted on Amazon S3. This website is available at the Amazon S3 website endpoint. However, you might have a domain, such as `example.com`, that you want to use to serve the content from the website you created. You might also want to use Amazon S3 root domain support to serve requests for both `http://www.example.com` and `http://example.com`. This requires additional steps. For an example, see Tutorial: Configuring a static website using a custom domain registered with Route 53.

Step 8: Clean up

If you created your static website only as a learning exercise, delete the AWS resources that you allocated so that you no longer accrue charges. After you delete your AWS resources, your website is no longer available. For more information, see Deleting a general purpose bucket.

Tutorial: Configuring a static website using a custom domain registered with Route 53

Suppose that you want to host a static website on Amazon S3. You've registered a domain with Amazon Route 53 (for example, `example.com`), and you want requests for `http://www.example.com` and `http://example.com` to be served from your Amazon S3 content. You can use this walkthrough to learn how to host a static website and create redirects on Amazon S3 for a website with a custom domain name that is registered with Route 53. You can work with an existing website that you want to host on Amazon S3, or use this walkthrough to start from scratch.

After you complete this walkthrough, you can optionally use Amazon CloudFront to improve the performance of your website. For more information, see Speeding up your website with Amazon CloudFront.

> **Note**
>
> Amazon S3 website endpoints do not support HTTPS or access points. If you want to use HTTPS, you can use Amazon CloudFront to serve a static website hosted on Amazon S3. For a tutorial about how to host your content securely with CloudFront and Amazon S3, see Tutorial: Hosting on-demand streaming video with Amazon S3, Amazon CloudFront, and Amazon Route 53. For more information, see How do I use CloudFront to serve a static website hosted on Amazon S3? and Requiring HTTPS for communication between viewers and CloudFront.

Automating static website setup with an CloudFormation template

You can use an CloudFormation template to automate your static website setup. The CloudFormation template sets up the components that you need to host a secure static website so that you can focus more on your website's content and less on configuring components.

The CloudFormation template includes the following components:

- Amazon S3 – Creates an Amazon S3 bucket to host your static website.
- CloudFront – Creates a CloudFront distribution to speed up your static website.
- Lambda@Edge – Uses Lambda@Edge to add security headers to every server response. Security headers are a group of headers in the web server response that tell web browsers to take extra security precautions. For more information, see the blog post Adding HTTP security headers using Lambda@Edge and Amazon CloudFront.

This CloudFormation template is available for you to download and use. For information and instructions, see Getting started with a secure static website in the *Amazon CloudFront Developer Guide*.

Topics

- Before you begin
- Step 1: Register a custom domain with Route 53
- Step 2: Create two buckets
- Step 3: Configure your root domain bucket for website hosting
- Step 4: Configure your subdomain bucket for website redirect
- Step 5: Configure logging for website traffic
- Step 6: Upload index and website content
- Step 7: Upload an error document
- Step 8: Edit S3 Block Public Access settings
- Step 9: Attach a bucket policy
- Step 10: Test your domain endpoint
- Step 11: Add alias records for your domain and subdomain
- Step 12: Test the website
- Speeding up your website with Amazon CloudFront
- Cleaning up your example resources

Before you begin

As you follow the steps in this example, you work with the following services:

Amazon Route 53 – You use Route 53 to register domains and to define where you want to route internet traffic for your domain. The example shows how to create Route 53 alias records that

route traffic for your domain (`example.com`) and subdomain (`www.example.com`) to an Amazon S3 bucket that contains an HTML file.

Amazon S3 – You use Amazon S3 to create buckets, upload a sample website page, configure permissions so that everyone can see the content, and then configure the buckets for website hosting.

Step 1: Register a custom domain with Route 53

If you don't already have a registered domain name, such as `example.com`, register one with Route 53. For more information, see Registering a new domain in the *Amazon Route 53 Developer Guide*. After you register your domain name, you can create and configure your Amazon S3 buckets for website hosting.

Step 2: Create two buckets

To support requests from both the root domain and subdomain, you create two buckets.

- **Domain bucket** – `example.com`
- **Subdomain bucket** – `www.example.com`

These bucket names must match your domain name exactly. In this example, the domain name is `example.com`. You host your content out of the root domain bucket (`example.com`). You create a redirect request for the subdomain bucket (`www.example.com`). If someone enters `www.example.com` in their browser, they are redirected to `example.com` and see the content that is hosted in the Amazon S3 bucket with that name.

To create your buckets for website hosting

The following instructions provide an overview of how to create your buckets for website hosting. For detailed, step-by-step instructions on creating a bucket, see Creating a general purpose bucket.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Create your root domain bucket:

    a. In the navigation bar, choose the name of the currently displayed AWS Region. Next, choose the Region in which you want to create a bucket.

> **Note**
>
> To minimize latency and costs and address regulatory requirements, choose a Region close to you. Objects stored in a Region never leave that Region unless you explicitly transfer them to another Region. For a list of Amazon S3 AWS Regions, see AWS service endpoints in the *Amazon Web Services General Reference*.

b. In the left navigation pane, choose **General purpose buckets**.

c. Choose **Create bucket**. The **Create bucket** page opens.

d. Enter the **Bucket name** (for example, **`example.com`**).

e. Choose the Region where you want to create the bucket.

Choose a Region that is geographically close to you to minimize latency and costs, or to address regulatory requirements. The Region that you choose determines your Amazon S3 website endpoint. For more information, see Website endpoints.

f. To accept the default settings and create the bucket, choose **Create**.

3. Create your subdomain bucket:

a. Choose **Create bucket**.

b. Enter the **Bucket name** (for example, **`www.example.com`**).

c. Choose the Region where you want to create the bucket.

Choose a Region that is geographically close to you to minimize latency and costs, or to address regulatory requirements. The Region that you choose determines your Amazon S3 website endpoint. For more information, see Website endpoints.

d. To accept the default settings and create the bucket, choose **Create**.

In the next step, you configure `example.com` for website hosting.

Step 3: Configure your root domain bucket for website hosting

In this step, you configure your root domain bucket (`example.com`) as a website. This bucket will contain your website content. When you configure a bucket for website hosting, you can access the website using the Website endpoints.

To enable static website hosting

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **General purpose buckets**.
3. In the buckets list, choose the name of the bucket that you want to enable static website hosting for.
4. Choose **Properties**.
5. Under **Static website hosting**, choose **Edit**.
6. Choose **Use this bucket to host a website**.
7. Under **Static website hosting**, choose **Enable**.
8. In **Index document**, enter the file name of the index document, typically `index.html`.

   The index document name is case sensitive and must exactly match the file name of the HTML index document that you plan to upload to your S3 bucket. When you configure a bucket for website hosting, you must specify an index document. Amazon S3 returns this index document when requests are made to the root domain or any of the subfolders. For more information, see Configuring an index document.
9. To provide your own custom error document for 4XX class errors, in **Error document**, enter the custom error document file name.

   The error document name is case sensitive and must exactly match the file name of the HTML error document that you plan to upload to your S3 bucket. If you don't specify a custom error document and an error occurs, Amazon S3 returns a default HTML error document. For more information, see Configuring a custom error document.
10. (Optional) If you want to specify advanced redirection rules, in **Redirection rules**, enter JSON to describe the rules.

    For example, you can conditionally route requests according to specific object key names or prefixes in the request. For more information, see Configure redirection rules to use advanced conditional redirects.
11. Choose **Save changes**.

    Amazon S3 enables static website hosting for your bucket. At the bottom of the page, under **Static website hosting**, you see the website endpoint for your bucket.
12. Under **Static website hosting**, note the **Endpoint**.

The **Endpoint** is the Amazon S3 website endpoint for your bucket. After you finish configuring your bucket as a static website, you can use this endpoint to test your website.

After you edit block public access settings and add a bucket policy that allows public read access, you can use the website endpoint to access your website.

In the next step, you configure your subdomain (`www.example.com`) to redirect requests to your domain (`example.com`).

Step 4: Configure your subdomain bucket for website redirect

After you configure your root domain bucket for website hosting, you can configure your subdomain bucket to redirect all requests to the domain. In this example, all requests for `www.example.com` are redirected to `example.com`.

To configure a redirect request

1. On the Amazon S3 console, in the **General purpose buckets** list, choose your subdomain bucket name (`www.example.com` in this example).
2. Choose **Properties**.
3. Under **Static website hosting**, choose **Edit**.
4. Choose **Redirect requests for an object**.
5. In the **Target bucket** box, enter your root domain, for example, **`example.com`**.
6. For **Protocol**, choose **http**.
7. Choose **Save changes**.

Step 5: Configure logging for website traffic

If you want to track the number of visitors accessing your website, you can optionally enable logging for your root domain bucket. For more information, see Logging requests with server access logging. If you plan to use Amazon CloudFront to speed up your website, you can also use CloudFront logging.

To enable server access logging for your root domain bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. In the same Region where you created the bucket that is configured as a static website, create a bucket for logging, for example `logs.example.com`.
3. Create a folder for the server access logging log files (for example, `logs`).
4. (Optional) If you want to use CloudFront to improve your website performance, create a folder for the CloudFront log files (for example, `cdn`).

   > **Important**
   >
   > When you create or update a distribution and enable CloudFront logging, CloudFront updates the bucket access control list (ACL) to give the `awslogsdelivery` account `FULL_CONTROL` permissions to write logs to your bucket. For more information, see Permissions required to configure standard logging and to access your log files in the *Amazon CloudFront Developer Guide*. If the bucket that stores the logs uses the Bucket owner enforced setting for S3 Object Ownership to disable ACLs, CloudFront cannot write logs to the bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

5. In the **Buckets** list, choose your root domain bucket.
6. Choose **Properties**.
7. Under **Server access logging**, choose **Edit**.
8. Choose **Enable**.
9. Under the **Target bucket**, choose the bucket and folder destination for the server access logs:

   - Browse to the folder and bucket location:
     1. Choose **Browse S3**.
     2. Choose the bucket name, and then choose the logs folder.
     3. Choose **Choose path**.
   - Enter the S3 bucket path, for example, `s3://logs.example.com/logs/`.

10. Choose **Save changes**.

    In your log bucket, you can now access your logs. Amazon S3 writes website access logs to your log bucket every 2 hours.

Step 6: Upload index and website content

In this step, you upload your index document and optional website content to your root domain bucket.

When you enable static website hosting for your bucket, you enter the name of the index document (for example, **`index.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this index document name to your bucket.

To configure the index document

1. Create an `index.html` file.

   If you don't have an `index.html` file, you can use the following HTML to create one:

   ```
   <html xmlns="http://www.w3.org/1999/xhtml" >
   <head>
       <title>My Website Home Page</title>
   </head>
   <body>
     <h1>Welcome to my website</h1>
     <p>Now hosted on Amazon S3!</p>
   </body>
   </html>
   ```

2. Save the index file locally.

   The index document file name must exactly match the index document name that you enter in the **Static website hosting** dialog box. The index document name is case sensitive. For example, if you enter `index.html` for the **Index document** name in the **Static website hosting** dialog box, your index document file name must also be `index.html` and not `Index.html`.

3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
4. In the left navigation pane, choose **General purpose buckets**.
5. In the buckets list, choose the name of the bucket that you want to use to host a static website.
6. Enable static website hosting for your bucket, and enter the exact name of your index document (for example, `index.html`). For more information, see Enabling website hosting.

After enabling static website hosting, proceed to step 6.

7. To upload the index document to your bucket, do one of the following:

   - Drag and drop the index file into the console bucket listing.
   - Choose **Upload**, and follow the prompts to choose and upload the index file.

   For step-by-step instructions, see Uploading objects.

8. (Optional) Upload other website content to your bucket.

Step 7: Upload an error document

When you enable static website hosting for your bucket, you enter the name of the error document (for example, **`404.html`**). After you enable static website hosting for the bucket, you upload an HTML file with this error document name to your bucket.

To configure an error document

1. Create an error document, for example `404.html`.
2. Save the error document file locally.

   The error document name is case sensitive and must exactly match the name that you enter when you enable static website hosting. For example, if you enter `404.html` for the **Error document** name in the **Static website hosting** dialog box, your error document file name must also be `404.html`.

3. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
4. In the left navigation pane, choose **General purpose buckets**.
5. In the buckets list, choose the name of the bucket that you want to use to host a static website.
6. Enable static website hosting for your bucket, and enter the exact name of your error document (for example, `404.html`). For more information, see Enabling website hosting and Configuring a custom error document.

   After enabling static website hosting, proceed to step 6.

7. To upload the error document to your bucket, do one of the following:

- Drag and drop the error document file into the console bucket listing.
- Choose **Upload**, and follow the prompts to choose and upload the index file.

For step-by-step instructions, see Uploading objects.

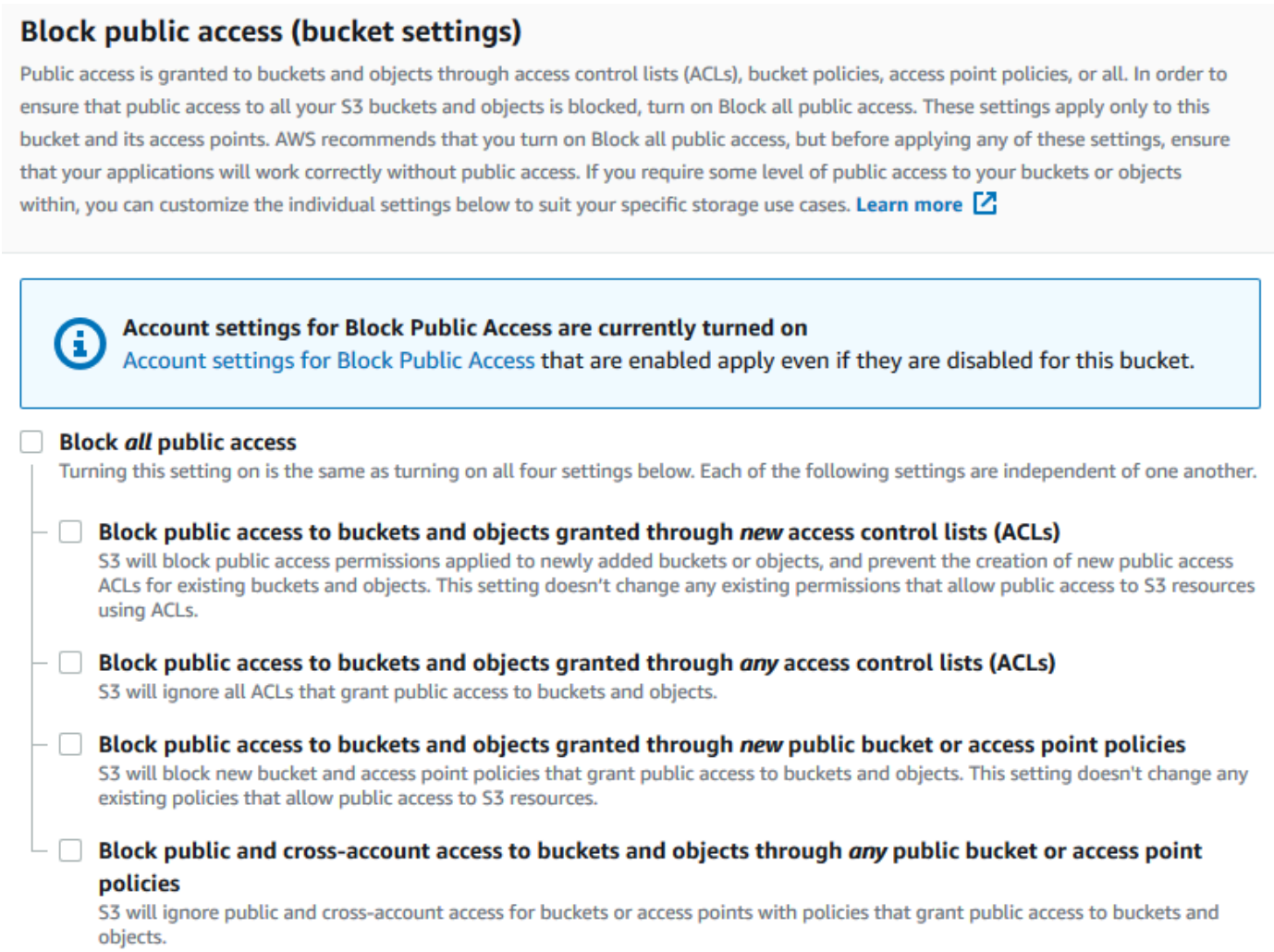
Step 8: Edit S3 Block Public Access settings

In this example, you edit block public access settings for the domain bucket (`example.com`) to allow public access.

By default, Amazon S3 blocks public access to your account and buckets. If you want to use a bucket to host a static website, you can use these steps to edit your block public access settings.

> **Warning**
>
> Before you complete these steps, review Blocking public access to your Amazon S3 storage to ensure that you understand and accept the risks involved with allowing public access. When you turn off block public access settings to make your bucket public, anyone on the internet can access your bucket. We recommend that you block all public access to your buckets.

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the bucket that you have configured as a static website.
3. Choose **Permissions**.
4. Under **Block public access (bucket settings)**, choose **Edit**.
5. Clear **Block *all* public access**, and choose **Save changes**.



Amazon S3 turns off the Block Public Access settings for your bucket. To create a public static website, you might also have to edit the Block Public Access settings for your account before adding a bucket policy. If the Block Public Access settings for your account are currently turned on, you see a note under **Block public access (bucket settings)**.

Step 9: Attach a bucket policy

In this example, you attach a bucket policy to the domain bucket (`example.com`) to allow public read access. You replace the *`Bucket-Name`* in the example bucket policy with the name of your domain bucket, for example `example.com`.

After you edit S3 Block Public Access settings, you can add a bucket policy to grant public read access to your bucket. When you grant public read access, anyone on the internet can access your bucket.

Important

The following policy is an example only and allows full access to the contents of your bucket. Before you proceed with this step, review How can I secure the files in my Amazon S3 bucket? to ensure that you understand the best practices for securing the files in your S3 bucket and risks involved in granting public access.

1. Under **Buckets**, choose the name of your bucket.
2. Choose **Permissions**.
3. Under **Bucket Policy**, choose **Edit**.
4. To grant public read access for your website, copy the following bucket policy, and paste it in the **Bucket policy editor**.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "PublicReadGetObject",
            "Effect": "Allow",
            "Principal": "*",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::Bucket-Name/*"
            ]
        }
    ]
}
```

5. Update the `Resource` to your bucket name.

   In the preceding example bucket policy, *Bucket-Name* is a placeholder for the bucket name. To use this bucket policy with your own bucket, you must update this name to match your bucket name.

6. Choose **Save changes**.

   A message appears indicating that the bucket policy has been successfully added.

If you see an error that says `Policy has invalid resource`, confirm that the bucket name in the bucket policy matches your bucket name. For information about adding a bucket policy, see How do I add an S3 bucket policy?

If you get an error message and cannot save the bucket policy, check your account and bucket Block Public Access settings to confirm that you allow public access to the bucket.

In the next step, you can figure out your website endpoints and test your domain endpoint.

Step 10: Test your domain endpoint

After you configure your domain bucket to host a public website, you can test your endpoint. For more information, see Website endpoints. You can only test the endpoint for your domain bucket because your subdomain bucket is set up for website redirect and not static website hosting.

> **Note**
>
> Amazon S3 does not support HTTPS access to the website. If you want to use HTTPS, you can use Amazon CloudFront to serve a static website hosted on Amazon S3.
> For more information, see How do I use CloudFront to serve a static website hosted on Amazon S3? and Requiring HTTPS for communication between viewers and CloudFront.

1. Under **Buckets**, choose the name of your bucket.
2. Choose **Properties**.
3. At the bottom of the page, under **Static website hosting**, choose your **Bucket website endpoint**.

   Your index document opens in a separate browser window.

In the next step, you use Amazon Route 53 to enable customers to use both of your custom URLs to navigate to your site.

Step 11: Add alias records for your domain and subdomain

In this step, you create the alias records that you add to the hosted zone for your domain maps `example.com` and `www.example.com`. Instead of using IP addresses, the alias records use the

Amazon S3 website endpoints. Amazon Route 53 maintains a mapping between the alias records and the IP addresses where the Amazon S3 buckets reside. You create two alias records, one for your root domain and one for your subdomain.

Add an alias record for your root domain and subdomain

To add an alias record for your root domain (`example.com`)

1. Open the Route 53 console at https://console.aws.amazon.com/route53/.

   > **Note**
   >
   > If you don't already use Route 53, see Step 1: Register a domain in the *Amazon Route 53 Developer Guide*. After completing your setup, you can resume the instructions.

2. Choose **Hosted zones**.
3. In the list of hosted zones, choose the name of the hosted zone that matches your domain name.
4. Choose **Create record**.
5. Choose **Switch to wizard**.

   > **Note**
   >
   > If you want to use quick create to create your alias records, see Configuring Route 53 to route traffic to an S3 Bucket.

6. Choose **Simple routing**, and choose **Next**.
7. Choose **Define simple record**.
8. In **Record name**, accept the default value, which is the name of your hosted zone and your domain.
9. In **Value/Route traffic to**, choose **Alias to S3 website endpoint**.
10. Choose the Region.
11. Choose the S3 bucket.

    The bucket name should match the name that appears in the **Name** box. In the **Choose S3 bucket** list, the bucket name appears with the Amazon S3 website endpoint for the Region

where the bucket was created, for example, `s3-website-us-west-1.amazonaws.com (example.com)`.

Choose S3 bucket lists a bucket if:

- You configured the bucket as a static website.
- The bucket name is the same as the name of the record that you're creating.
- The current AWS account created the bucket.

If your bucket does not appear in the **Choose S3 bucket** list, enter the Amazon S3 website endpoint for the Region where the bucket was created, for example, **`s3-website-us-west-2.amazonaws.com`**. For a complete list of Amazon S3 website endpoints, see Amazon S3 Website endpoints. For more information about the alias target, see Value/route traffic to in the *Amazon Route 53 Developer Guide*.

12. In **Record type**, choose **A - Routes traffic to an IPv4 address and some AWS resources**.
13. For **Evaluate target health**, choose **No**.
14. Choose **Define simple record**.

To add an alias record for your subdomain (`www.example.com`)

1. Under **Configure records**, choose **Define simple record**.
2. In **Record name** for your subdomain, type `www`.
3. In **Value/Route traffic to**, choose **Alias to S3 website endpoint**.
4. Choose the Region.
5. Choose the S3 bucket, for example, `s3-website-us-west-2.amazonaws.com (www.example.com)`.

   If your bucket does not appear in the **Choose S3 bucket** list, enter the Amazon S3 website endpoint for the Region where the bucket was created, for example, **`s3-website-us-west-2.amazonaws.com`**. For a complete list of Amazon S3 website endpoints, see Amazon S3 Website endpoints. For more information about the alias target, see Value/route traffic to in the *Amazon Route 53 Developer Guide*.
6. In **Record type**, choose **A - Routes traffic to an IPv4 address and some AWS resources**.
7. For **Evaluate target health**, choose **No**.
8. Choose **Define simple record**.

9. On the **Configure records** page, choose **Create records**.

> **Note**
>
> Changes generally propagate to all Route 53 servers within 60 seconds. When propagation is done, you can route traffic to your Amazon S3 bucket by using the names of the alias records that you created in this procedure.

Add an alias record for your root domain and subdomain (old Route 53 console)

To add an alias record for your root domain (`example.com`)

The Route 53 console has been redesigned. In the Route 53 console you can temporarily use the old console. If you choose to work with the old Route 53 console, use the procedure below.

1. Open the Route 53 console at https://console.aws.amazon.com/route53/.

   > **Note**
   >
   > If you don't already use Route 53, see Step 1: Register a domain in the *Amazon Route 53 Developer Guide*. After completing your setup, you can resume the instructions.

2. Choose **Hosted Zones**.
3. In the list of hosted zones, choose the name of the hosted zone that matches your domain name.
4. Choose **Create Record Set**.
5. Specify the following values:

   Name

   Accept the default value, which is the name of your hosted zone and your domain.

   For the root domain, you don't need to enter any additional information in the **Name** field.

   Type

   Choose **A – IPv4 address**.

Alias

Choose **Yes**.

Alias Target

In the **S3 website endpoints** section of the list, choose your bucket name.

The bucket name should match the name that appears in the **Name** box. In the **Alias Target** listing, the bucket name is followed by the Amazon S3 website endpoint for the Region where the bucket was created, for example `example.com (s3-website-us-west-2.amazonaws.com)`. **Alias Target** lists a bucket if:

- You configured the bucket as a static website.
- The bucket name is the same as the name of the record that you're creating.
- The current AWS account created the bucket.

If your bucket does not appear in the **Alias Target** listing, enter the Amazon S3 website endpoint for the Region where the bucket was created, for example, `s3-website-us-west-2`. For a complete list of Amazon S3 website endpoints, see Amazon S3 Website endpoints. For more information about the alias target, see Value/route traffic to in the *Amazon Route 53 Developer Guide*.

Routing Policy

Accept the default value of **Simple**.

Evaluate Target Health

Accept the default value of **No**.

6. Choose **Create**.

To add an alias record for your subdomain (`www.example.com`)

1. In the hosted zone for your root domain (`example.com`), choose **Create Record Set**.
2. Specify the following values:

Name

For the subdomain, enter `www` in the box.

Type

Choose **A – IPv4 address**.

Alias

Choose **Yes**.

Alias Target

In the **S3 website endpoints** section of the list, choose the same bucket name that appears in the **Name** field—for example, `www.example.com (s3-website-us-west-2.amazonaws.com)`.

Routing Policy

Accept the default value of **Simple**.

Evaluate Target Health

Accept the default value of **No**.

3. Choose **Create**.

> **Note**
>
> Changes generally propagate to all Route 53 servers within 60 seconds. When propagation is done, you can route traffic to your Amazon S3 bucket by using the names of the alias records that you created in this procedure.

Step 12: Test the website

Verify that the website and the redirect work correctly. In your browser, enter your URLs. In this example, you can try the following URLs:

- **Domain** (`http://example.com`) – Displays the index document in the `example.com` bucket.
- **Subdomain** (`http://www.example.com`) – Redirects your request to `http://example.com`. You see the index document in the `example.com` bucket.

If your website or redirect links don't work, you can try the following:

- **Clear cache** – Clear the cache of your web browser.
- **Check name servers** – If your web page and redirect links don't work after you've cleared your cache, you can compare the name servers for your domain and the name servers for your hosted zone. If the name servers don't match, you might need to update your domain name servers to match those listed under your hosted zone. For more information, see Adding or changing name servers and glue records for a domain.

After you've successfully tested your root domain and subdomain, you can set up an Amazon CloudFront distribution to improve the performance of your website and provide logs that you can use to review website traffic. For more information, see Speeding up your website with Amazon CloudFront.

Speeding up your website with Amazon CloudFront

You can use Amazon CloudFront to improve the performance of your Amazon S3 website. CloudFront makes your website files (such as HTML, images, and video) available from data centers around the world (known as *edge locations*). When a visitor requests a file from your website, CloudFront automatically redirects the request to a copy of the file at the nearest edge location. This results in faster download times than if the visitor had requested the content from a data center that is located farther away.

CloudFront caches content at edge locations for a period of time that you specify. If a visitor requests content that has been cached for longer than the expiration date, CloudFront checks the origin server to see if a newer version of the content is available. If a newer version is available, CloudFront copies the new version to the edge location. Changes that you make to the original content are replicated to edge locations as visitors request the content.

Using CloudFront without Route 53

The tutorial on this page uses Route 53 to point to your CloudFront distribution. However, if you want to serve content hosted in an Amazon S3 bucket using CloudFront without using Route 53, see Amazon CloudFront Tutorials: Setting up a Dynamic Content Distribution for Amazon S3. When you serve content hosted in an Amazon S3 bucket using CloudFront, you can use any bucket name, and both HTTP and HTTPS are supported.

Automating set up with an CloudFormation template

For more information about using an CloudFormation template to configure a secure static website that creates a CloudFront distribution to serve your website, see Getting started with a secure static website in the *Amazon CloudFront Developer Guide*.

Topics

- Step 1: Create a CloudFront distribution
- Step 2: Update the record sets for your domain and subdomain
- (Optional) Step 3: Check the log files

Step 1: Create a CloudFront distribution

First, you create a CloudFront distribution. This makes your website available from data centers around the world.

To create a distribution with an Amazon S3 origin

1. Open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. Choose **Create Distribution**.
3. On the **Create Distribution** page, in the **Origin Settings** section, for **Origin Domain Name**, enter the Amazon S3 website endpoint for your bucket—for example, **`example.com.s3-website.us-west-1.amazonaws.com`**.

   CloudFront fills in the **Origin ID** for you.
4. For **Default Cache Behavior Settings**, keep the values set to the defaults.

   With the default settings for **Viewer Protocol Policy**, you can use HTTPS for your static website. For more information these configuration options, see Values that You Specify When You Create or Update a Web Distribution in the *Amazon CloudFront Developer Guide*.
5. For **Distribution Settings**, do the following:

   a. Leave **Price Class** set to **Use All Edge Locations (Best Performance)**.

   b. Set **Alternate Domain Names (CNAMEs)** to the root domain and `www` subdomain. In this tutorial, these are `example.com` and `www.example.com`.

> **Important**
>
> Before you perform this step, note the requirements for using alternate domain names, in particular the need for a valid SSL/TLS certificate.

c. For **SSL Certificate**, choose **Custom SSL Certificate (example.com)**, and choose the custom certificate that covers the domain and subdomain names.

   For more information, see SSL Certificate in the *Amazon CloudFront Developer Guide*.

d. In **Default Root Object**, enter the name of your index document, for example, `index.html`.

   If the URL used to access the distribution doesn't contain a file name, the CloudFront distribution returns the index document. The **Default Root Object** should exactly match the name of the index document for your static website. For more information, see Configuring an index document.

e. Set **Logging** to **On**.

> **Important**
>
> When you create or update a distribution and enable CloudFront logging, CloudFront updates the bucket access control list (ACL) to give the `awslogsdelivery` account `FULL_CONTROL` permissions to write logs to your bucket. For more information, see Permissions required to configure standard logging and to access your log files in the *Amazon CloudFront Developer Guide*. If the bucket that stores the logs uses the Bucket owner enforced setting for S3 Object Ownership to disable ACLs, CloudFront cannot write logs to the bucket. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.

f. For **Bucket for Logs**, choose the logging bucket that you created.

   For more information about configuring a logging bucket, see (Optional) Logging web traffic.

g. If you want to store the logs that are generated by traffic to the CloudFront distribution in a folder, in **Log Prefix**, enter the folder name.

h. Keep all other settings at their default values.

6. Choose **Create Distribution**.
7. To see the status of the distribution, find the distribution in the console and check the **Status** column.

   A status of `InProgress` indicates that the distribution is not yet fully deployed.

   After your distribution is deployed, you can reference your content with the new CloudFront domain name.
8. Record the value of **Domain Name** shown in the CloudFront console, for example, `dj4p1rv6mvubz.cloudfront.net`.
9. To verify that your CloudFront distribution is working, enter the domain name of the distribution in a web browser.

   If your website is visible, the CloudFront distribution works. If your website has a custom domain registered with Amazon Route 53, you will need the CloudFront domain name to update the record set in the next step.

Step 2: Update the record sets for your domain and subdomain

Now that you have successfully created a CloudFront distribution, update the alias record in Route 53 to point to the new CloudFront distribution.

To update the alias record to point to a CloudFront distribution

1. Open the Route 53 console at https://console.aws.amazon.com/route53/.
2. In the left navigation, choose **Hosted zones**.
3. On the **Hosted Zones** page, choose the hosted zone that you created for your subdomain, for example, `www.example.com`.
4. Under **Records**, select the *A* record that you created for your subdomain.
5. Under **Record details**, choose **Edit record**.
6. Under **Route traffic to**, choose **Alias to CloudFront distribution**.
7. Under **Choose distribution**, choose the CloudFront distribution.
8. Choose **Save**.
9. To redirect the *A* record for the root domain to the CloudFront distribution, repeat this procedure for the root domain, for example, `example.com`.

The update to the record sets takes effect within 2–48 hours.

10. To see whether the new *A* records have taken effect, in a web browser, enter your subdomain URL, for example, `http://www.example.com`.

    If the browser no longer redirects you to the root domain (for example, `http://example.com`), the new A records are in place. When the new *A* record has taken effect, traffic routed by the new *A* record to the CloudFront distribution is not redirected to the root domain. Any visitors who reference the site by using `http://example.com` or `http://www.example.com` are redirected to the nearest CloudFront edge location, where they benefit from faster download times.

    > **Tip**
    >
    > Browsers can cache redirect settings. If you think the new *A* record settings should have taken effect, but your browser still redirects `http://www.example.com` to `http://example.com`, try clearing your browser history and cache, closing and reopening your browser application, or using a different web browser.

(Optional) Step 3: Check the log files

The access logs tell you how many people are visiting the website. They also contain valuable business data that you can analyze with other services, such as Amazon EMR.

CloudFront logs are stored in the bucket and folder that you choose when you create a CloudFront distribution and enable logging. CloudFront writes logs to your log bucket within 24 hours from when the corresponding requests are made.

To see the log files for your website

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the logging bucket for your website.
3. Choose the CloudFront logs folder.
4. Download the `.gzip` files written by CloudFront before opening them.

   If you created your website only as a learning exercise, you can delete the resources that you allocated so that you no longer accrue charges. To do so, see Cleaning up your example resources. After you delete your AWS resources, your website is no longer available.

Cleaning up your example resources

If you created your static website as a learning exercise, you should delete the AWS resources that you allocated so that you no longer accrue charges. After you delete your AWS resources, your website is no longer available.

Tasks

- Step 1: Delete the Amazon CloudFront distribution
- Step 2: Delete the Route 53 hosted zone
- Step 3: Disable logging and delete your S3 bucket

Step 1: Delete the Amazon CloudFront distribution

Before you delete an Amazon CloudFront distribution, you must disable it. A disabled distribution is no longer functional and does not accrue charges. You can enable a disabled distribution at any time. After you delete a disabled distribution, it is no longer available.

To disable and delete a CloudFront distribution

1. Open the CloudFront console at https://console.aws.amazon.com/cloudfront/v4/home.
2. Select the distribution that you want to disable, and then choose **Disable**.
3. When prompted for confirmation, choose **Yes, Disable**.
4. Select the disabled distribution, and then choose **Delete**.
5. When prompted for confirmation, choose **Yes, Delete**.

Step 2: Delete the Route 53 hosted zone

Before you delete the hosted zone, you must delete the record sets that you created. You don't need to delete the NS and SOA records; these are automatically deleted when you delete the hosted zone.

To delete the record sets

1. Open the Route 53 console at https://console.aws.amazon.com/route53/.
2. In the list of domain names, select your domain name, and then choose **Go to Record Sets**.
3. In the list of record sets, select the *A* records that you created.

The type of each record set is listed in the **Type** column.

4. Choose **Delete Record Set**.
5. When prompted for confirmation, choose **Confirm**.

To delete a Route 53 hosted zone

1. Continuing from the previous procedure, choose **Back to Hosted Zones**.
2. Select your domain name, and then choose **Delete Hosted Zone**.
3. When prompted for confirmation, choose **Confirm**.

Step 3: Disable logging and delete your S3 bucket

Before you delete your S3 bucket, make sure that logging is disabled for the bucket. Otherwise, AWS continues to write logs to your bucket as you delete it.

To disable logging for a bucket

1. Open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Under **Buckets**, choose your bucket name, and then choose **Properties**.
3. From **Properties**, choose **Logging**.
4. Clear the **Enabled** check box.
5. Choose **Save**.

Now, you can delete your bucket. For more information, see Deleting a general purpose bucket.

Deploying a static website to AWS Amplify Hosting from an S3 general purpose bucket

We recommend that you use AWS Amplify Hosting to host static website content stored on S3. Amplify Hosting is a fully managed service that makes it easy to deploy your websites on a globally available content delivery network (CDN) powered by Amazon CloudFront, allowing secure static website hosting without extensive setup. With AWS Amplify Hosting, you can select the location of your objects within your general purpose bucket, deploy your content to a managed CDN, and generate a public HTTPS URL for your website to be accessible anywhere. Deploying a static website using Amplify Hosting provides you with the following benefits and features:

- **Deployment to the AWS content delivery network (CDN) powered by Amazon CloudFront** - CloudFront is a web service that speeds up distribution of your static and dynamic web content to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the request is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance, increased reliability and availability. For more information, see How CloudFront delivers content in the *Amazon CloudFront Developer Guide*.
- **HTTPS support** - Provides secure communication and data transfer between your website and a user's web browser.
- **Custom domains** - Easily connect your website to a custom URL purchased from a domain registrar such as Amazon Route 53.
- **Custom SSL certificates** - When you set up your custom domain, you can use the default managed certificate that Amplify provisions for you or you can use your own custom certificate purchased from the third-party certificate authority of your choice.
- **Built in metrics and CloudWatch monitoring** - Monitor traffic, errors, data transfer, and latency for your website.
- **Password protection** - Restrict access to your website, by setting up a username and password requirement in the Amplify console.
- **Redirects and rewrites** - Create redirect and rewrite rules in the Amplify console to enable a web server to reroute navigation from one URL to another.

When you deploy your application from an Amazon S3 general purpose bucket to Amplify Hosting, AWS charges are based on Amplify's pricing model. For more information, see AWS Amplify Pricing.

Important

Amplify Hosting is not available in all of the AWS Regions where Amazon S3 is available. To deploy a static website to Amplify Hosting, the Amazon S3 general purpose bucket containing your website must be located in a region where Amplify is available. For the list of regions where Amplify is available, see Amplify endpoints in the *Amazon Web Services General Reference*.

You can start the deployment process from the Amazon S3 console, the Amplify console, the AWS CLI, or the AWS SDKs. You can only deploy to Amplify from a general purpose bucket located in your own account. Amplify doesn't support cross-account bucket access.

Use the following instructions to deploy a static website from an Amazon S3 general purpose bucket to Amplify Hosting starting from the Amazon S3 console.

Deploying a static website to Amplify from the S3 console

To deploy a static website from the Amazon S3 console

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **Buckets**.
3. In the **Buckets** list, choose the general purpose bucket that contains the website you want to deploy to Amplify Hosting.
4. Choose the **Properties** tab.
5. Under **Static website hosting**, choose **Create Amplify app**. At this step, the deployment process will move to the Amplify console.
6. On the **Deploy with S3** page, do the following steps.
   a. For **App name**, enter the name of your app or website.
   b. For **Branch name**, enter the name of your app's backend.
   c. For **S3 location of objects to host**, either enter the directory path to your general purpose bucket or choose **Browse S3** to locate and select it.
7. Choose **Save and deploy**.

> **Note**
>
> If you update any of the objects for a static website in your general purpose bucket hosted on Amplify, you must redeploy the application to Amplify Hosting to cause the changes to take effect. Amplify Hosting doesn't automatically detect changes to your bucket. For more information, see Updating a static website deployed to Amplify from an S3 bucket in the *AWS Amplify Hosting User Guide*.

To start directly from the Amplify console, see Deploying a static website from S3 using the Amplify console in the *AWS Amplify Hosting User Guide*.

To get started using the AWS SDKs, see Creating a bucket policy to deploy a static website from S3 using the AWS SDKs in the *AWS Amplify Hosting User Guide*.

Tagging for cost allocation or attribute-based access control (ABAC)

An AWS tag is a key-value pair that holds metadata. You attach the tags to your Amazon S3 resources, such as buckets. You can tag resources when you create them or manage tags on existing resources. There is no additional charge for using tags beyond the standard S3 API request rates. For more information, see Amazon S3 pricing.

How tags work

Amazon S3 support two types of tags:

- **AWS-generated tags:** AWS automatically applies these tags, and you cannot modify them or remove them. To learn more about AWS-generated tags, see Using AWS-generated tags.
- **User-defined tags:** You apply these tags to your S3 resources and manage them.

User-defined tags

A user-defined tag is a tag key-value pair that you use to label your resources. User-defined tags consist of a required key and an optional value. These are the main components of a user-defined tag:

The tag key

The tag key is the required name of the tag. For example, `project` is the tag key in the following tag key-value pair:

Key	Value
`project`	`Trinity`

The tag key is a case-sensitive string that must contain between 1 and 128 Unicode characters. Keys can only contain Unicode letters or numbers, white space, and the following symbols:

- _ - underscore

- `.` - period
- `:` - colon
- `/` - forward slash
- `=` - equal sign
- `+` - plus sign
- `@` - at sign
- `-` - dash

The tag value

The tag value is an optional string. For example, `Trinity` is the tag value in this tag key-value pair:

Key	Value
`project`	`Trinity`

The tag value is a case-sensitive string that can contain between 0 and 256 Unicode characters. Values can only contain Unicode letters or numbers, white space, and the following symbols:

- `_` - underscore
- `.` - period
- `:` - colon
- `/` - forward slash
- `=` - equal sign
- `+` - plus sign
- `@` - at sign
- `-` - dash

For details on the characters allowed in user-defined tags and other restrictions, see User-Defined Tag Restrictions in the *AWS Billing and Cost Management User Guide*.

The tag set

Each S3 resource has one *tag set* that contains all of the tag key and value pairs that are assigned to that bucket. A tag set can be empty, or it can contain as many as 50 user-defined tags, except in the case of tagging objects. Objects can have up to 10 tags only.

While each key must be unique in a tag set, you can use the same value multiple times. For example, you can have the same value, `Trinity`, for following two tag keys:

Key	Value
`project`	`Trinity`
`cost-center`	`Trinity`

When you add a tag that has the same key as an existing tag, the new value overwrites the old value.

AWS does not apply any semantic meaning to your tags. We interpret tags strictly as character strings.

To add, list, modify, or delete tags, you can use the Amazon S3 console, the AWS Command Line Interface (AWS CLI), or the Amazon S3 API.

Common ways to use tags

Use tags on your S3 resources for:

1. **Cost allocation** – Track storage costs by bucket tag in AWS Billing and Cost Management.
2. **Attribute-based access control (ABAC)** – Scale access permissions and grant access to S3 resources based on their tags.

> **Note**
>
> You can use the same tags for both cost allocation and access control.

Using tags for cost allocation

Track your Amazon S3 storage costs by applying tags to S3 resources and activating these tags for cost allocation.

To start tracking costs:

1. Add tags to your S3 resources or use existing tags. For example, you can label buckets with a tag that identifies a project or a group of projects.
2. Activate the tags for cost allocation in the AWS Billing and Cost Management console. See Activating user-defined cost allocation tags in the *AWS Billing and Cost Management User Guide*. You can activate user-defined or AWS-generated tags. For more information, see Organizing and tracking costs using AWS cost allocation tags.

AWS uses the activated tags to organize your resource costs in various billing and cost management tools, such as:

- **Cost allocation report**

  Provides a high-level view of costs organized by your activated tags. For more information, see Using the monthly cost allocation report in the *AWS Billing User Guide*.
- **Cost and Usage Report (CUR)**

  Provides the most detailed set of AWS cost and usage data, including tag-based cost breakdowns. For more information, see What are AWS Cost and Usage Reports? in the *AWS Data Export User Guide*.

Amazon S3 resources that support tracking costs with tags

The following Amazon S3 resources support storage cost tracking using tags.

- **S3 general purpose buckets**

  For more information, see Using tags with S3 general purpose buckets.
- **S3 Directory buckets**

  For more information, see Using tags with S3 directory buckets.
- **S3 tables**

For more information, see Using tags with S3 tables.

- **S3 vector index**

  For more information, see Using tags with S3 vector indexes

Using tags for attribute-based access control (ABAC)

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes, i.e., tags. You can attach tags to AWS Identity and Access Management (IAM) entities (users or roles) and to AWS resources, such as Amazon S3 Access Points and directory buckets. Then, you control permissions to these resources using tag-based conditions in access control policies to allow or deny operations when these conditions are met.

With ABAC, you use tag-based condition keys in your AWS organizations, IAM, and S3 resource policies. For enterprises, ABAC in Amazon S3 supports authorization across multiple AWS accounts.

In your IAM policies, you can control access to S3 resources based on the bucket's tags by using condition keys.

Supported condition keys

You can use the following AWS condition keys for all Amazon S3 resources that support ABAC.

- `aws:ResourceTag/key-name`
- `aws:RequestTag/key-name`
- `aws:TagKeys`

Some resources support additional Amazon S3 condition keys. For a complete list of condition keys that can be used for ABAC and example policies, see the following tagging topics for the resource.

Amazon S3 resources that support ABAC

The following Amazon S3 resources support attribute-based access control (ABAC) using tags.

- **S3 general purpose buckets**

  For more information, see Using tags with S3 general purpose buckets.

- **Access points**

  For more information, see Using tags with S3 Access Points for general purpose buckets and Using tags with S3 Access Points for directory buckets.

- **Batch operation jobs**

  For more information, see Controlling access and labeling jobs using tags.

- **S3 Directory buckets**

  For more information, see Using tags with S3 directory buckets.

- **Objects**

  For more information, see Tagging your objects.

- **S3 Access Grants**

  For more information, see Managing tags for S3 Access Grants.

- **S3 Storage Lens Groups**

  For more information, see Managing AWS resource tags with Storage Lens groups.

- **S3 table buckets and tables**

  For more information, see Using tags with S3 table buckets and Using tags with S3 tables.

- **S3 Vector buckets and indexes**

  For more information, see Using tags with S3 vector buckets and Using tags with S3 vector indexes.

Planning your tagging strategy

We recommend that you devise a set of tag keys that meets your needs for each resource type. Using a consistent set of tag keys makes it easier for you to manage your resources. You can search and filter the resources based on the tags you add. For more information about how to implement an effective resource tagging strategy, see the Tagging Best Practices AWS Whitepaper.

Best practices for tagging Amazon S3 resources

When you use tags, we recommend that you adhere to the following best practices:

- Document conventions for tag use that are followed by all teams in your organization. In particular, ensure the names are both descriptive and consistent. For example, standardize on the format `environment:prod` rather than tagging some resources with `env:production`.

  > **Important**
  >
  > Do not store personally identifiable information (PII) or other confidential or sensitive information in tags.

- Automate tagging to ensure consistency. For example, you can include tags in an CloudFormation template. When you create resources with the template, the resources are tagged automatically.
- Use tags only when necessary. Avoid unnecessary tag proliferation and complexity.
- Review tags periodically for relevance and accuracy. Remove or modify outdated tags as needed.
- Consider creating tags with the AWS Tag Editor in the AWS Management Console. You can use the Tag Editor to add tags to multiple supported AWS resources, including Amazon S3 resources, at the same time. For more information, see Tag Editor in the *AWS Resource Groups User Guide*.

Managing tags for Amazon S3 resources

For more information on how to manage tags for Amazon S3 resources, see the following:

- Using tags with S3 Access Points for general purpose buckets
- Using tags with S3 Access Points for directory buckets
- Controlling access and labeling jobs using tags
- Using tags with S3 directory buckets
- Using cost allocation S3 bucket tags
- Tagging your objects
- Managing tags for S3 Access Grants
- Managing AWS resource tags with Storage Lens groups
- Using tags with S3 general purpose buckets.
- Using tags with S3 vector buckets
- Using tags with S3 vector indexes

Using tags with S3 general purpose buckets

An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 general purpose buckets. You can tag S3 buckets when you create them or manage tags on existing buckets. For general information about tags, see Tagging for cost allocation or attribute-based access control (ABAC).

> **Note**
>
> There is no additional charge for using tags on buckets beyond the standard S3 API request rates. For more information, see Amazon S3 pricing.

Common ways to use tags with buckets

Use tags on your S3 buckets for:

1. **Cost allocation** – Track storage costs by bucket tag in AWS Billing and Cost Management. For more information, see Using tags for cost allocation.
2. **Attribute-based access control (ABAC)** – Scale access permissions and grant access to S3 buckets based on their tags. For more information, see Using tags for ABAC.

> **Note**
>
> For general purpose buckets, ABAC is not enabled by default. To enable ABAC for general purpose buckets, see Enabling ABAC in general purpose buckets. For Amazon S3 resources such as access points and directory buckets, ABAC is enabled by default. You can use the same tags for both cost allocation and access control.

ABAC for S3 general purpose buckets

Amazon S3 general purpose buckets support attribute-based access control (ABAC) using tags. Use tag-based condition keys in your AWS organizations, IAM, and S3 bucket policies. For enterprises, ABAC in Amazon S3 supports authorization across multiple AWS accounts.

In your IAM policies, you can control access to S3 buckets based on the bucket's tags by using the following global condition keys:

- `aws:ResourceTag/key-name`
  - Use this condition key to compare the tag key-value pair that you specify in the policy with the key-value pair attached to the resource. S3 evaluates this condition key only after you enable ABAC on your bucket. For example, you could require that access to a resource is allowed only if the resource has the attached tag key `Dept` with the value `Marketing`. For more information, see Controlling access to AWS resources.
- `aws:RequestTag/key-name`
  - Use this condition key to compare the tag key-value pair that was passed in the request with the tag pair that you specify in the policy. For example, you could check whether the request includes the tag key `Dept` and that it has the value `Accounting`. For more information, see Controlling access during AWS requests. You can use this condition key to restrict which tag key-value pairs can be passed during the `TagResource` and `CreateBucket` API operations.
- `aws:TagKeys`
  - Use this condition key to compare the tag keys in a request with the keys that you specify in the policy. We recommend that when you use policies to control access using tags, use the `aws:TagKeys` condition key to define what tag keys are allowed. For example policies and more information, see Controlling access based on tag keys.
- `s3:BucketTag/tag-key`
  - Use this condition key to grant permissions to specific data in buckets using tags. This condition key is applicable only after ABAC is enabled on your bucket. When accessing a bucket by using an access point, the `aws:ResourceTag/tag-key` condition key references the tags on the bucket both when authorizing against the access point and the bucket. The `s3:BucketTag/tag-key` will reference the tags only of the bucket it is being authorized against.

> **Note**
>
> When creating buckets with tags, note that tag-based conditions to access your bucket using aws:ResourceTag and s3:BucketTag condition keys are applicable only after you enable ABAC on the bucket. To learn more, see Enabling ABAC in general purpose buckets.

Example ABAC policies for buckets

See the following example ABAC policies for Amazon S3 buckets.

1.1 - IAM policy to create or modify buckets with specific tags

In this IAM policy, users or roles with this policy can only create S3 buckets if they tag the bucket with the tag key `project` and tag value `Trinity` in the bucket creation request. They can also add or modify tags on existing S3 buckets as long as the `TagResource` request includes the tag key-value pair `project:Trinity`. This policy does not grant read, write, or delete permissions on the buckets or its objects.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateBucketWithTags",
      "Effect": "Allow",
      "Action": [
        "s3:CreateBucket",
        "s3:TagResource"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/project": [
            "Trinity"
          ]
        }
      }
    }
  ]
}
```

1.2 - Bucket policy to restrict operations

In this bucket policy, IAM principals (users and roles) are denied `s3:ListBucket`, `s3:GetObject`, and `s3:PutObject` actions on the bucket only if value of the `project` tag on the bucket matches the value of the `project` tag on the principal.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "DenyObjectOperations",
            "Effect": "Deny",
            "Principal": "*",
            "Action": ["s3:ListBucket",
                       "s3:GetObject",
                       "s3:PutObject"],
            "Resource": "arn:aws:s3:::aws-s3-demo/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/project": "${aws:PrincipalTag/project}"
                }
            }
        }
    ]
}
```

1.3 - IAM policy to modify tags on existing resources

In this IAM policy, IAM principals (users or roles) can modify tags on a bucket only if the value of the bucket's `project` tag matches the value of the principal's `project` tag. Only the four tags `project`, `environment`, `owner`, and `cost-center` specified in the `aws:TagKeys` condition keys are permitted for these buckets. This helps enforce tag governance, prevents unauthorized tag modifications, and keeps the tagging schema consistent across your buckets.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceTaggingRulesOnModification",
      "Effect": "Allow",
      "Action": [
        "s3:TagResource",
        "s3:CreateBucket"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringEquals": {
          "aws:TagKeys": [
            "project",
            "environment",
            "owner",
```

```
            "cost-center"
          ]
        }
      }
    }
  ]
}
```

1.4 - Using the s3:BucketTag condition key

In this IAM policy, the condition statement allows access to the `aws-s3-demo` bucket's data only if the bucket has the tag key `Environment` and tag value `Production`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessViaSpecificBucket",
      "Effect": "Allow",
      "Action": "*",
      "Resource": ["arn:aws:s3:::aws-s3-demo","arn:aws:s3:::aws-s3-demo/*"],
      "Condition": {
        "StringEquals": {
          "s3:BucketTag/Environment": "Production"
        }
      }
    }
  ]
}
```

Managing tags for general purpose buckets

You can add or manage tags for S3 buckets using the Amazon S3 Console, the AWS Command Line Interface (CLI), the AWS SDKs, or using the S3 APIs: TagResource, UntagResource, and ListTagsForResource. For more information, see:

Enabling ABAC in general purpose buckets

Attribute-based access control (ABAC) is an authorization strategy that you use to define permissions based on attributes, i.e., tags. By default, ABAC is disabled for all Amazon S3 general purpose buckets. To use ABAC for general purpose buckets, you must enable it.

Before enabling ABAC for your general purpose bucket, we recommend that you first complete the tasks described in the following topics:

Topics

- Auditing your policies before enabling ABAC

Auditing your policies before enabling ABAC

Before you enable ABAC for your bucket, if your bucket has tags, audit your access control policies to review if tag-based conditions reference any of the existing tags on your buckets. If they do, confirm that these policies are set up as intended and that enabling tag-based access control does not create unintentional authorization changes to your Amazon S3 workflows. Doing so will help you ensure that your policies function as intended after ABAC is enabled on your buckets. For examples of using attribute-based conditions with tags, see Using tags with S3 general purpose buckets.

Including the required permissions in your IAM policies

You need the following Amazon S3 permissions to enable ABAC for your bucket:

- `s3:PutBucketAbac` – Update the ABAC status for your general purpose bucket.
- `s3:GetBucketAbac` – View the ABAC status for your general purpose bucket

After you enable ABAC, the permissions you previously used to add tags to a bucket or delete tags from a bucket, `PutBucketTagging` or `DeleteBucketTagging`, will no longer work. Instead, use the `TagResource` and *`UntagResource`* APIs to perform these tasks.

We recommend you use `TagResource` and `UntagResource` APIs to manage tagging before enabling ABAC on your buckets. The Amazon S3 Console and CloudFormation now use the `TagResource` and `UntagResource` APIs by default. You can also disable ABAC on your bucket by using the `PutBucketAbac` API. You can use `GetBucketTagging` to list tags on your buckets. This API will continue to work after you enable ABAC for your buckets. Alternatively you can also use `ListTagsForResource` to list all tags on your buckets.

You will need the following permissions to apply tags to and remove them from general purpose buckets.

- `s3:TagResource` - Add tags to an AWS resource, such as an Amazon S3 general purpose bucket.
- `s3:UntagResource` - Remove tags from an AWS resource, such as an Amazon S3 general purpose bucket.
- `s3:ListTagsForResource` - View the tags applied to an AWS resource, such as an Amazon S3 general purpose bucket.

The following IAM policy grants the permission to enable ABAC and view its status for your bucket.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutBucketAbac",
        "s3:GetBucketAbac"
      ],
      "Resource": "arn:aws:s3:::my-s3-bucket/*"
    }
  ]
}
```

For more information on tagging general purpose buckets and example ABAC policies for general purpose buckets, see Using tags with S3 general purpose buckets.

Steps

If you have `s3:PutBucketAbac` permission for a general purpose bucket, you can enable ABAC for the bucket by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To enable ABAC for a general purpose bucket using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **buckets**.
3. Choose the bucket name.

4. Choose the **Properties** tab.
5. In the **Bucket ABAC** panel, and choose **Edit**.
6. Choose the **Enable** toggle.
7. Review and acknowledge the permissions you will need to manage tags after you enable ABAC: `TagResource`, `UntagResource`, and `ListTagsForResource`.
8. Choose **Save changes**.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to add enable ABAC for a general purpose bucket by using the AWS SDK for Java 2.x. To use the command replace the *`user input placeholders`* with your own information.

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbacStatus;
import software.amazon.awssdk.services.s3.model.GetBucketAbacRequest;
import software.amazon.awssdk.services.s3.model.GetBucketAbacResponse;
import software.amazon.awssdk.services.s3.model.PutBucketAbacRequest;
import software.amazon.awssdk.services.s3.model.PutBucketAbacResponse;
import software.amazon.awssdk.regions.Region;

public class BucketAbac {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putBucketAbac(s3, "amzn-s3-demo-bucket", "Enabled");
        getBucketAbac(s3, "amzn-s3-demo-bucket");

        putBucketAbac(s3, "amzn-s3-demo-bucket", "Disabled");
        getBucketAbac(s3, "amzn-s3-demo-bucket");
    }

    /**
     * Sets the ABAC (Attribute-Based Access Control) status for a specified S3
 bucket.
     *
```

```
     * @param s3 The S3Client instance to use for the API call
     * @param bucketName The name of the S3 bucket to update
     * @param status The desired ABAC status ("Enabled" or "Disabled")
    */
    public static void putBucketAbac(S3Client s3, String bucketName, String status)
 {
       try {
            AbacStatus abacStatus = AbacStatus.builder()
                .abacStatus(status)
                .build();
            PutBucketAbacReqquest request = PutBucketAbacRequest.builder()
                .bucket(bucketName)
                .abacStatus(abacStatus)
                .build();
            s3.putBucketAbac(request);
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the current ABAC (Attribute-Based Access Control) status for a
 specified S3 bucket.
     *
     * @param s3 The S3Client instance to use for the API call
     * @param bucketName The name of the S3 bucket to query
    */
    public static void getBucketAbac(S3Client s3, String bucketName) {
       try {
            GetBucketAbacReqquest request = GetBucketAbacRequest.builder()
                .bucket(bucketName)
                .build();
            GetBucketAbacResponse response = s3.getBucketAbac(request);
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

This example shows you how to add enable ABAC for a general purpose bucket by using the AWS SDK for Java 2.x. To use the command replace the `user input placeholders` with your own information.

Using the REST API

For information about the Amazon S3 REST API support for adding tags to a general purpose bucket, see the following section in the *Amazon Simple Storage Service API Reference*:

- PutBucketAbac

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to enable ABAC for a general purpose bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
# Enable ABAC on a general purpose bucket

aws s3api put-bucket-abac --bucket amzn-s3-demo-bucket --abac-status Status=Enabled --
region us-east-2

# Disable ABAC on a general purpose bucket

aws s3api put-bucket-abac --bucket amzn-s3-demo-bucket --abac-status Status=Disabled --
region us-east-2

# Get ABAC status on a general purpose bucket

aws s3api get-bucket-abac --bucket amzn-s3-demo-bucket --region us-east-2
```

Creating general purpose buckets with tags

You can tag Amazon S3 general purpose buckets when you create them. There is no additional charge for using tags on buckets beyond the standard S3 API request rates. For more information,

see Amazon S3 pricing. For more information about tagging buckets, see Using tags with S3 general purpose buckets.

Permissions

To create a bucket with tags, you must have the following permissions:

- `s3:CreateBucket`
- `s3:TagResource`

Amazon S3 Console and CloudFormation now use this capability to create buckets with tags.

Troubleshooting errors

If you encounter an error when attempting to create a bucket with tags, you can do the following:

- Verify that you have the required Permissions to create the bucket and add a tag to it.
- Check your IAM policy for `aws:TagKeys` or `aws:RequestTag/key-name` condition keys. You might be required to label your buckets only with specific tag keys and values. For more information, see Using tags for attribute-based access control (ABAC).

> **Note**
>
> Amazon S3 and CloudFormation now use this capability to create buckets with tags. When creating buckets with tags, note that tag-based conditions to access your bucket using `aws:ResourceTag` and `s3:BucketTag` condition keys are applicable only after you enable ABAC on the bucket. To learn more, see Enabling ABAC in general purpose buckets.

Steps

You can create a bucket with tags applied by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To create a bucket with tags using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **buckets**.
3. Choose **create bucket** to create a new bucket.
4. Create a bucket a general purpose bucket as you normally would; see Creating a general purpose bucket.
5. On the **Create bucket** page, **Tags** is an option when creating a new bucket.
6. Enter a name for the bucket. For more information, see General purpose bucket naming rules.
7. Choose **Add new Tag** to open the Tags editor and enter a tag key-value pair. The tag key is required, but the value is optional.
8. To add another tag, select **Add new Tag** again. You can enter up to 50 tag key-value pairs.
9. After you complete specifying the options for your new bucket, choose **Create bucket**.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to create a general purpose bucket with tags by using the AWS SDK for Java 2.x. To use the command replace the *`user input placeholders`* with your own information.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLocationConstraint;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.Tag;

public class CreateBucketWithTagsExample {
    public static void createBucketWithTagsExample() {
        S3Client s3 = S3Client.builder().region(Region.US_WEST_2).build();

        CreateBucketConfiguration bucketConfiguration =
 CreateBucketConfiguration.builder()
                .locationConstraint(BucketLocationConstraint.US_WEST_2)
                .tags(Tag.builder().key("MyTagKey").value("MyTagValue").build())
                .build();
```

```
        CreateBucketRequest createBucketRequest = CreateBucketRequest.builder()
                .bucket("mybucket")
                .createBucketConfiguration(bucketConfiguration)
                .build();

        CreateBucketResponse response = s3.createBucket(createBucketRequest);
        System.out.println("Status code (should be 200):");
        System.out.println(response.sdkHttpResponse().statusCode());
    }
}
```

Using the REST API

For information about the Amazon S3 REST API support for creating a general purpose bucket with tags, see the following section in the *Amazon Simple Storage Service API Reference*:

- CreateBucket

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to create a bucket with tags by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

When you create a bucket, you must provide configuration details and use the following naming convention: *`amzn-s3-demo-bucket`*

Request:

```
aws s3api create-bucket \
--bucket mybucket \
--create-bucket-configuration 'LocationConstraint=us-
west-2,Tags=[{Key=MyTagKey,Value=MyTagValue}]' --region us-west-2"
```

Response:

```
{
  "Location": "http://mybuckets3.amazonaws.com/"
}
```

Adding a tag to a bucket

You can add tags to Amazon S3 buckets and modify these tags. There is no additional charge for using tags on buckets beyond the standard S3 API request rates. For more information, see Amazon S3 pricing. For more information about tagging buckets, see Using tags with S3 general purpose buckets.

Permissions

To add a tag to a bucket, you must have the following permission:

- `s3:TagResource`

Troubleshooting errors

If you encounter an error when attempting to add a tag to a bucket, you can do the following:

- Verify that you have the required Permissions to add a tag to a bucket.
- If you attempted to add a tag key that starts with the AWS reserved prefix `aws:`, change the tag key and try again.

Steps

You can add tags to buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To add tags to a bucket using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **buckets**.
3. Choose the bucket name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and choose **Add new Tag**.
6. This opens the **Add Tags** page. You can enter up to 50 tag key value pairs.

7. If you add a new tag with the same key name as an existing tag, the value of the new tag overrides the value of the existing tag.
8. You can also edit the values of existing tags on this page.
9. After you have added the tag(s), choose **Save changes**.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to add tags to a general purpose bucket by using the AWS SDK for Java 2.x. To use the command replace the *user input placeholders* with your own information.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.Tag;
import software.amazon.awssdk.services.s3control.model.TagResourceRequest;
import software.amazon.awssdk.services.s3control.model.TagResourceResponse;

public class TagResourceExample {
    public static void tagResourceExample() {
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                .resourceArn("arn:aws::s3:::bucket/my-bucket")
                .accountId("111122223333")
                .tags(Tag.builder().key("MyTagKey").value("MyTagValue").build())
                .build();

        TagResourceResponse response = s3Control.tagResource(tagResourceRequest);
        System.out.println("Status code (should be 204):");
        System.out.println(response.sdkHttpResponse().statusCode());
    }
}
```

Using the REST API

For information about the Amazon S3 REST API support for adding tags to a general purpose bucket, see the following section in the *Amazon Simple Storage Service API Reference*:

- TagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to add tags to a general purpose bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws s3control tag-resource --resource-arn arn:aws:s3:::amzn-s3-demo-bucket --region us-east-2 --account-id 444455556666 --tags '[{"Key":"mykey","Value":"myvalue"}]'
```

Viewing bucket tags

You can view or list tags applied to S3 buckets. For more information about tags, see Using tags with S3 general purpose buckets.

Permissions

To view tags applied to a bucket, you must have the following permission:

- `s3:ListTagsForResource`

Troubleshooting errors

If you encounter an error when attempting to list or view the tags of a bucket, you can do the following:

- Verify that you have the required Permissions to view or list the tags of the bucket.

Steps

You can view tags applied to buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To view tags applied to a bucket using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **buckets**.
3. Choose the bucket name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section to view all of the tags applied to the general purpose bucket.
6. The **Tags** section shows the **User-defined tags** by default. You can select the **AWS-generated tags** tab to view tags applied to your bucket by AWS services.

Using the AWS SDKs

This section provides an example of how to view tags applied to a general purpose bucket by using the AWS SDKs.

SDK for Java 2.x

This example shows you how to view tags applied to a general purpose bucket by using the AWS SDK for Java 2.x.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceRequest;
import software.amazon.awssdk.services.s3control.model.ListTagsForResourceResponse;

public class ListTagsForResourceExample {
    public static void listTagsForResourceExample() {
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        ListTagsForResourceRequest listTagsForResourceRequest =
 ListTagsForResourceRequest.builder()
                .resourceArn("arn:aws::s3:::bucket/my-bucket")
                .accountId("111122223333")
                .build();

        ListTagsForResourceResponse response =
 s3Control.listTagsForResource(listTagsForResourceRequest);
```

```
        System.out.println("Tags on my resource:");
        System.out.println(response.toString());
    }
}
```

Using the REST API

For information about the Amazon S3 REST API support for viewing the tags applied to a general purpose bucket, see the following section in the *Amazon Simple Storage Service API Reference*:

- ListTagsforResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to view tags applied to a bucket. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws s3control list-tags-for-resource \
--account-id 111122223333 \
--resource-arn arn:aws::s3:us-east-1:444455556666:bucket/prefix--use1-az4--x-s3 \
```

Response - tags present:

```
{
  "Tags": [
      {
          "Key": "MyKey1",
          "Value": "MyValue1"
      },
      {
          "Key": "MyKey2",
          "Value": "MyValue2"
      },
      {
          "Key": "MyKey3",
          "Value": "MyValue3"
```

```
      }
  ]
}
```

Response - no tags present:

```
{
  "Tags": []
}
```

Deleting a tag from a bucket

You can remove tags from S3 buckets. An AWS tag is a key-value pair that holds metadata about resources, in this case Amazon S3 buckets. For more information about tags, see Using tags with S3 general purpose buckets.

> **Note**
>
> If you delete a tag and later learn that it was being used to track costs or for access control, you can add the tag back to the bucket.

Permissions

To delete a tag from a bucket, you must have the following permission:

- `s3:UntagResource`

Troubleshooting errors

If you encounter an error when attempting to delete a tag from a bucket, you can do the following:

- Verify that you have the required Permissions to delete a tag from a bucket.

Steps

You can delete tags from buckets by using the Amazon S3 console, the AWS Command Line Interface (AWS CLI), the Amazon S3 REST API, and AWS SDKs.

Using the S3 console

To delete tags from a bucket using the Amazon S3 console:

1. Sign in to Amazon S3 console at https://console.aws.amazon.com/s3/.
2. In the left navigation pane, choose **buckets**.
3. Choose the bucket name.
4. Choose the **Properties** tab.
5. Scroll to the **Tags** section and select the checkbox next to the tag or tags that you would like to delete.
6. Choose **Delete**.
7. The **Delete user-defined tags** pop-up appears and asks you to confirm the deletion of the tag or tags you selected.
8. Choose **Delete** to confirm.

Using the AWS SDKs

SDK for Java 2.x

This example shows you how to delete tags from a general purpose bucket by using the AWS SDK for Java 2.x. To use the command replace the *`user input placeholders`* with your own information.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlClient;
import software.amazon.awssdk.services.s3control.model.UntagResourceRequest;
import software.amazon.awssdk.services.s3control.model.UntagResourceResponse;

public class UntagResourceExample {
    public static void untagResourceExample() {
        S3ControlClient s3Control =
 S3ControlClient.builder().region(Region.US_WEST_2).build();

        UntagResourceRequest untagResourceRequest = UntagResourceRequest.builder()
                .resourceArn("arn:aws::s3:::bucket/my-bucket")
                .accountId("111122223333")
                .tagKeys("myTagKey")
                .build();
```

```
        UntagResourceResponse response =
 s3Control.untagResource(untagResourceRequest);
        System.out.println("Status code (should be 204):");
        System.out.println(response.sdkHttpResponse().statusCode());
    }
}
```

Using the REST API

For information about the Amazon S3 REST API support for deleting tags from a general purpose bucket, see the following section in the *Amazon Simple Storage Service API Reference*:

- UnTagResource

Using the AWS CLI

To install the AWS CLI, see Installing the AWS CLI in the *AWS Command Line Interface User Guide*.

The following CLI example shows you how to delete tags from a general purpose bucket by using the AWS CLI. To use the command replace the *`user input placeholders`* with your own information.

Request:

```
aws s3control untag-resource \
--resource-arn arn:aws::s3:::amzn-s3-demo-bucket --region us-east-2 --account-
id 111122223333 \
--tag-keys "tagkey1" "tagkey2"
```

Quotas

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Amazon S3 quotas include number of general purpose buckets, directory buckets, access points and more. You can request an increase for some quotas, but not all quotas can be increased. These increases are not granted immediately, so it may take a couple of days for your increase to become effective.

For a list of Amazon S3 quotas and their default values see, Amazon S3 quotas in the *AWS General Reference*.

Quota increases

To request a quota increase

You can request a quota increase by using one of following options:

- From the AWS Management Console: Open the Service Quotas console. In the navigation pane, choose **AWS services**. Select **Amazon S3**, select a quota, and follow the directions to request a quota increase. For instructions, see Requesting a quota increase in the *Service Quotas User Guide*.
- From the AWS CLI: Use the request-service-quota-increase AWS CLI command. For instructions, see Requesting a quota increase in the *Service Quotas User Guide*.

Reference

In addition to the Amazon S3 console, you can work with Amazon S3 programmatically using the Amazon S3 REST APIs, AWS SDKs, or AWS Command Line Interface (AWS CLI).

- Amazon Simple Storage Service API Reference
- Code examples in the *Amazon Simple Storage Service API Reference*
- AWS Command Line Interface in the *AWS Command Line Interface User Guide*.
- AWS SDKs

SDK documentation	Code examples
AWS SDK for C++	AWS SDK for C++ code examples
AWS CLI	AWS CLI code examples.
AWS SDK for Go	AWS SDK for Go code examples
AWS SDK for Java	AWS SDK for Java code examples
AWS SDK for JavaScript	AWS SDK for JavaScript code examples
AWS SDK for Kotlin	AWS SDK for Kotlin code examples
AWS SDK for .NET	AWS SDK for .NET code examples
AWS SDK for PHP	AWS SDK for PHP code examples
AWS Tools for PowerShell	Tools for PowerShell code examples
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) code examples
AWS SDK for Ruby	AWS SDK for Ruby code examples
AWS SDK for Rust	AWS SDK for Rust code examples
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP code examples
AWS SDK for Swift	AWS SDK for Swift code examples

Document history

- **Current API version:** 2006-03-01

The following table describes the important changes in each release of the *Amazon Simple Storage Service API Reference* and the *Amazon S3 User Guide*. For notification about updates to this documentation, you can subscribe to an RSS feed.

Change	Description	Date
S3 Tables now supports migrating existing data to tables	You can now migrate existing Parquet data from AWS Glue Data Catalog Hive tables in Amazon S3 general purpose buckets into S3 Tables as query-ready Apache Iceberg tables, without provisioning compute. For more informati on, see Migrating tables to S3 Tables.	November 4, 2026
S3 Tables increases the default table bucket quota	The default quota for the number of Amazon S3 table buckets you can create per AWS Region in an account increased from 10 to 100. For more information, see S3 Tables Regions, endpoints, and service quotas.	September 30, 2026
Amazon S3 Express One Zone adds new AWS managed policies	Amazon S3 Express One Zone added new AWS-managed policies called `AmazonS3ExpressFul lAccess` and `AmazonS3E xpressReadOnlyAcce`	April 3, 2026

	ss . For more information, see AWS managed policies for Amazon S3 Express One Zone.	
S3 Tables updated the AmazonS3TablesFull Access AWS managed policy for S3 Tables	S3 Tables updated the AWS-managed policy called AmazonS3TablesFull Access . This policy now grants permissions to pass a role to the S3 Tables Replicati on service. For more informati on, see AWS managed policies for S3 Tables.	December 2, 2025
S3 Tables now supports automatic replication	S3 Tables now supports automatic replication of Apache Iceberg tables stored in Amazon S3 table buckets. For more information, see Replicating S3 tables.	December 2, 2025
S3 Tables now supports Intelligent-Tiering access tiers	S3 Tables now supports Intelligent-Tiering access tiers which continuously monitors access patterns and automatically moves table data between access tiers. For more information, see Cost optimization for tables with Intelligent-Tiering.	December 2, 2025

Large object support	Amazon S3 has increased the maximum size of an object you can store in an S3 bucket from 5 TB to 50 TB. For uploading these larger objects, you must implement the Multipart Upload REST API to upload objects in parts, and when downloading these objects, use concurrent `GetObject` requests, as single GET requests are limited to 5 TB. For more information, see Uploading objects and Downloading objects.	December 2, 2025
Amazon S3 Metadata is now available in 22 additional AWS Regions	For more information, see Amazon S3 Metadata AWS Regions.	November 26, 2025

Amazon S3 console adds integration with Amazon SageMaker Unified Studio for Amazon S3 Tables	You can now query your data in S3 Tables by using a rich SQL editor provided by Amazon SageMaker Unified Studio, accessible directly from the S3 console with a few clicks. Within SageMaker Unified Studio, you can write SQL queries or ask questions using natural language, and a SQL query is automatic ally generated based on the context of your data. Then, you can schedule queries to automate data processing. For more information, see Using S3 Tables with SageMaker Unified Studio.	November 21, 2025

Amazon S3 now supports tags on S3 Tables for attribute-based access control (ABAC) and cost allocation	Amazon S3 now supports tagging S3 Tables for attribute-based access control (ABAC) and cost allocation. You can use tags for ABAC to automatically manage permissions for users and roles accessing table buckets and tables, eliminating frequent IAM or S3 Tables resource-based policy updates and simplifying how you govern access at scale. You can also add tags to individua l tables to track and organize AWS costs using AWS Billing and Cost Management. Amazon S3 supports tags on S3 Tables in all AWS Regions where S3 Tables is available . You can get started with tagging by using the AWS Management Console, SDK, API, or CLI. For more informati on, see Tagging for cost allocation or attribute-based access control (ABAC).	November 6, 2025
Amazon S3 Metadata is now available in three additional AWS Regions	Amazon S3 Metadata is now available in three additiona l AWS Regions: Europe (Frankfurt), Europe (Ireland) , and Asia Pacific (Tokyo). For more information, see Amazon S3 Metadata AWS Regions.	October 22, 2025

S3 Object Lambda will no longer be open to new customers starting on November 7th, 2025	S3 Object Lambda will no longer be open to new customers starting November 7, 2025. If you would like to use S3 Object Lambda, sign up prior to that date. Existing customers can continue to use the service as normal. For more information, see Amazon S3 Object Lambda availability change.	October 7, 2025

Amazon S3 now supports conditional deletes in general purpose buckets	Amazon S3 now supports conditional deletes in general purpose buckets. Conditional deletes verify that an object is unchanged before deleting it. Using conditional deletes helps you to prevent accidental deletions in high-concurrency, multiple-writer scenarios. You can now perform conditional deletes using the HTTP `if-match` header with an ETag value. Amazon S3 only allows your delete request to succeed if the Etag provided matches that of the object. Additionally, you can use the `s3:if-match` condition key in your S3 bucket policies to enforce conditional delete operations. For example, you can require clients to use the HTTP `if-match` header in both `DeleteObject` and `DeleteObjects` API operation requests, helping you to minimize the risk of accidentally deleting objects in your bucket. For more information, see How to perform conditional deletes.	September 16, 2025

Amazon S3 Batch Operations now supports managing objects within an S3 bucket, prefix, suffix, or more, in a single step in the Amazon S3 console	When creating an S3 Batch Operations job, you can specify the objects on which to perform the operation. With this feature, you can instead specify an entire bucket, prefix, suffix, creation date, or storage class. Batch Operations will then apply the operation to all the matching objects and notify you when the job is finished. For more information, see Performing object operations in bulk with Batch Operations.	September 15, 2025
Amazon S3 now supports Access Points tags for attribute-based access control (ABAC)	Amazon S3 now supports tagging Access Points for both general purpose and directory buckets. Add tags to access points and use attribute-based access control (ABAC) to scale access. You can get started with tagging access points using the AWS Management Console, S3 REST API, AWS CLI, or the AWS SDK. To learn more about using tags for ABAC, see Using tags for attribute-based access control (ABAC).	July 31, 2025

Amazon S3 Vectors (Preview)	Amazon S3 Vectors delivers purpose-built, cost-optimized vector storage for your semantic search and AI applications. S3 Vectors introduces a new bucket type, the vector bucket, which is optimized for durable, low-cost vector storage. For more information, see Working with S3 Vectors and vector buckets.	July 15, 2025

Amazon S3 Metadata now supports existing objects	Amazon S3 Metadata creates and manages metadata that provides information about datasets stored in S3, so you can more easily discover and use your S3 data. Previously, S3 Metadata supported new and updated objects. Now, S3 Metadata also creates and manages metadata for all your existing S3 data. S3 Metadata now creates and maintains two Apache Iceberg tables that give you a qu eryable and continuously updated view of the metadata associated with the objects in a general purpose bucket. A journal table records changes made to your data in near real time, helping you to identify new data uploaded to your bucket, track recently deleted objects, monitor S3 Lifecycle transitions, and more. A new live inventory table provides and maintains an up-to-dat e view of all objects in your bucket. Both S3 Metadata tables are stored in an AWS managed Iceberg table bucket in your account, and can be queried using standard SQL through AWS analytics services, such as Amazon Athena or Amazon EMR, and	July 15, 2025

	open source tooling, such as DuckDB and PyIceberg. For more information, see Accelerating data discovery with S3 Metadata.	
Amazon S3 Express One Zone now supports tags for cost allocation and attribute-based access control	Amazon S3 Express One Zone, a high performance S3 storage class, now supports tags for cost allocation and attribute-based access control (ABAC). You can add tags to S3 directory buckets to track and organize AWS costs using AWS Billing and Cost Management. Additionally, with ABAC support, you can scale access control to your directory buckets by using tags. S3 Express One Zone supports tags on directory buckets in all AWS Regions where the storage class is available. You can get started with tagging using the AWS Management Console, S3 REST API, AWS CLI, or the AWS SDK. To learn more about using tags to simplify cost allocation or ABAC, see Using tags with S3 directory buckets.	July 2, 2025

S3 Tables now supports CloudTrail events for maintenance operations	S3 Tables now supports AWS CloudTrail events for maintenance operation s. You can now track and audit maintenance activities performed on your S3 Tables resources through CloudTrai l logs. For more information, see S3 Tables maintenance.	June 30, 2025
Access points now supports attaching to an FSx for OpenZFS volume	You can create and attach an access point to an FSx for OpenZFS volume using the Amazon FSx console, AWS CLI, or API. Once attached, you can use the S3 object APIs to access your file data. Your data continues to reside on the Amazon FSx file system and continues to be directly accessible for your existing workloads. For more information, see Managing access to shared datasets with access points.	June 25, 2025
S3 Tables now supports new compaction strategies for tables	S3 Tables now supports multiple compaction strategie s to optimize query perform ance based on different data access patterns. The new strategies include Sort and Z-order. For more information, see Compaction Strategies for tables.	June 18, 2025

Enhanced access denied error messages for same-organization requests	Amazon S3 now includes additional context in access denied (HTTP `403 Forbidden`) errors for requests made to resources within the same organization in AWS Organizations. This new context includes the type of policy that denied access, the reason for denial, and information about the AWS Identity and Access Management (IAM) user or role that requested access to the resource. This context helps you to troubleshoot access issues, identify the root cause of access denied errors, and fix incorrect access controls by updating the relevant policies. This additional context is also available in AWS CloudTrail logs. Enhanced access denied error messages for same-organization requests are now available in all AWS Regions, including the AWS GovCloud (US) Regions and the China Regions. For more information, see Troubleshoot access denied (403 Forbidden) errors in Amazon S3..	June 16, 2025

S3 Tables adds new AWS managed policy for S3 Tables	S3 Tables added a new AWS-managed policy called `AmazonS3TablesLakeFormationServiceRole`. This policy grants permissions that allows the Lake Formation service role access to Amazon S3 tables and table buckets. For more information, see AWS managed policies for S3 Tables.	May 19, 2025
S3 Tables now supports SSE-KMS using customer managed keys	S3 Tables now supports server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) using customer managed keys. You can apply SSE-KMS encryption at the table bucket and table level. For more information, see Using server-side encryption with AWS KMS keys (SSE-KMS) in table buckets.	April 16, 2025

Access points for directory buckets are available in AWS Local Zones	Directory buckets now support access points to simplify managing data access at scale for shared datasets in Amazon S3. With this new feature, you can create hundreds of access points per bucket, each with a distinct name and permissions customized for each applicati on. For more information, see Managing access to shared datasets in directory buckets with access points.	March 31, 2025

S3 table bucket integration with Amazon SageMaker Lakehouse is now generally available	You can integrate S3 table buckets with Amazon SageMaker Lakehouse to access tables from AWS analytics services, such as Amazon Athena, Amazon Redshift, and Quick. Amazon SageMaker Lakehouse unifies your data across Amazon S3 data lakes and Amazon Redshift data warehouses, so you can build analytics, machine learning (ML), and generative AI applications on a single copy of data. The integration populates the AWS Glue Data Catalog with your table resources , and federates access to these resources with AWS Lake Formation. The integration enables fine-grained access control through Lake Formation to provide additional security. For more information about integrating, see Using Amazon S3 Tables with AWS analytics services. If you set up the integration with the preview release, you can continue to use your current integration. However, the updated integration process provides performance improvements, so we recommend migrating	March 13, 2025

	. To migrate to the updated integration, see Migrating to the updated integration process.	
S3 Tables create table and query table support added to the Amazon S3 console	S3 Tables now support create table and query table operations directly from the Amazon S3 console by using Amazon Athena. With this new feature, you can now create a table, populate it with data, and query it with just a few steps in the Amazon S3 console. For more information, see Creating an Amazon S3 table and Querying Amazon S3 tables with Athena.	March 13, 2025
S3 Tables adds support for adding schemas when creating tables	You can now use the S3 Tables `CreateTable` API operation to create tables with schemas by adding an optional metadata flag. For more information, see Creating an Amazon S3 table.	January 30, 2025

S3 Metadata is now generally available	Amazon S3 Metadata is now generally available. S3 Metadata helps you easily discover and understand your S3 data with automated, queryable metadata that updates in near real-time. With S3 Metadata, you can curate, identify, and use your S3 data for business analytics, artificial intelligence and machine learning (AI/ML) model training, and more. S3 Metadata supports object metadata, which includes system-defined information like the size and the source of the object, and custom metadata, such as tags with information like product SKUs, transaction IDs, or content ratings. For more information, see Accelerating data discovery with S3 Metadata.	January 27, 2025
AWS managed policies – New policies	S3 Tables added two new AWS managed policies.	December 6, 2024

S3 Tables	Amazon S3 Tables provide S3 storage that's optimized for analytics workloads, with features that improve query performance, reduce storage costs for tables, and simplify the operation of data lakes at scale. S3 Tables introduces a new bucket type: table buckets, which are purpose-built for storing Apache Iceberg tables as subresources. Table buckets provide higher transactions per second (TPS) and better query throughput compared to self-managed tables in S3 general purpose buckets. You can automatically integrate your table buckets with AWS analytics services, such as Athena, Amazon Redshift Quick, and more. For more information, see Working with Amazon S3 Tables and table buckets.	December 3, 2024

S3 Metadata preview	Amazon S3 Metadata helps you easily discover and understand your S3 data with automated, queryable metadata that updates in near real-time. With S3 Metadata, you can curate, identify, and use your S3 data for business analytics, artificial intelligence and machine learning (AI/ML) model training, and more. S3 Metadata supports object metadata, which includes system-defined information like the size and the source of the object, and custom metadata, such as tags with information like product SKUs, transaction IDs, or content ratings. For more information, see Accelerating data discovery with S3 Metadata.	December 3, 2024
Storage Browser for S3	Storage Browser for S3 is an open source component that you can add to your web applications to provide your end users with a simple interface for data stored in S3. For more information, see Storage Browser for S3.	December 1, 2024

New Amazon S3 checksum algorithm and improved checksum integrity features	Amazon S3 adds the `CRC-64NVME` checksum algorithm and improved checksum integrity features. For more information, see Checking object integrity in Amazon S3.	December 1, 2024
Data residency workloads	In Dedicated Local Zones, you can create S3 directory buckets to store data for your data residency and isolation use cases. For more information, see Data residency workloads.	December 1, 2024
New condition keys to enforce conditional writes	Amazon S3 adds new condition keys `s3:if-match` and `s3:if-none-match` to use in bucket policies to force clients to use the `If-None-Match` or `If-Match` HTTP header. For more information, see Enforce conditional writes on Amazon S3 buckets.	November 25, 2024

New HTTP header for conditional writes to check if the object has changed	Amazon S3 adds the `If-Match` HTTP header to check an object's entity tag (ETag) before writing an object for some API operations. With this header, Amazon S3 compares the provided ETag value with the ETag value of the object in S3. If the ETag values don't match, the operation fails. For more information, see How to prevent object overwrites with conditional writes.	November 25, 2024
Amazon Redshift now integrates with S3 Access Grants	Amazon Redshift customers can now use S3 Access Grants to scale and manage permissions for their S3 data. This allows Amazon Redshift customers to scale S3 permissions for corporate identities by using AWS IAM Identity Center as well as for IAM users and groups. For more information, see Amazon Redshift integration with Amazon S3 Access Grants.	November 15, 2024

AWS Organizations member accounts can now regain access to accidentally locked Amazon S3 buckets	AWS Organizations member accounts can now use a simple process through AWS Identity and Access Management (IAM) to regain access to accidentally locked Amazon S3 buckets. For more information, see Perform a privileged task on an AWS Organizations member account in the *AWS Identity and Access Management User Guide*.	November 14, 2024
Resource control policies (RCPs), a new type of authorization policy in AWS Organizations is also available for Amazon S3 buckets	Resource control policies (RCPs), a new authorization policy managed in AWS Organizations can be used to set the maximum available permissions on Amazon S3 buckets within your entire organization. For more information, see Resource control policies (RCPs) in the *AWS Organizations User Guide*.	November 13, 2024
Amazon S3 now automatically approves bucket quota increases up to 1,000 buckets	Amazon S3 now automatically approves bucket quota increases up to 1,000 buckets. To view your bucket utilization or request an increase, visit the Service Quotas console.	September 30, 2024

New default minimum object size transition behavior for Amazon S3 Lifecycle configurations	Amazon S3 now applies a default behavior to S3 Lifecycle configurations that prevents objects smaller than 128 KB from being transitioned to any storage class. To learn how to override this behavior, see Allowing objects smaller than 128 KB to be transitioned.	September 24, 2024
Directory buckets now support SSE-KMS using customer managed keys.	Directory buckets now supports server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) using customer managed keys. You have more options to encrypt and manage the security of your data in directory buckets. For more information, see Data protection and encryption in directory buckets.	September 17, 2024

S3 Access Grants supports an API action that lists the caller's access grants	IAM identities and IAM Identity Center corporate directory identities can now use the `ListCallerAccessGrants` API to list all of the Amazon S3 buckets, prefixes, and objects they can access, as defined by their S3 Access Grants. Use this API to discover all of the S3 data an IAM or corporate directory identity can access through S3 Access Grants, within a particular AWS account. For more information, see List the caller's access grants.	September 5, 2024

Enhanced access denied error messages for same-account requests	Amazon S3 now includes additional context in access denied (HTTP `403 Forbidden`) errors for requests made to resources within the same AWS account. This new context includes the type of policy that denied access, the reason for denial, and information about the AWS Identity and Access Management (IAM) user or role that requested access to the resource. This context helps you to troubleshoot access issues, identify the root cause of access denied errors, and fix incorrect access controls by updating the relevant policies. This additional context is also available in AWS CloudTrail logs. Enhanced access denied error messages for same-account requests are now available in all AWS Regions, including the AWS GovCloud (US) Regions and the China Regions. For more information, see Troubleshoot access denied (403 Forbidden) errors in Amazon S3..	August 21, 2024

Amazon S3 supports using conditional writes for `PutObject` and `CompleteMultipartUpload`	You can check for the existence of an object in your bucket before creating it using a conditional write on upload operations. This can prevent overwrites of existing data. Conditional writes will validate there is no existing object with the same key name already in your bucket. For more information, see Conditional requests.	August 20, 2024
Amazon S3 no longer charges for several HTTP error codes	Amazon S3 has completed a change so unauthorized requests that customers did not initiate are free of charge. For more information, see Billing for Amazon S3 error responses.	August 19, 2024
Amazon S3 Select is no longer available to new customers	Amazon S3 Select is no longer available to new customers. Existing customers of Amazon S3 Select can continue to use the feature as usual. Learn more.	July 25, 2024

Amazon S3 Inventory supports the s3:InventoryAccessibleOptionalFields condition key	Amazon S3 Inventory supports the s3:InventoryAccessibleOptionalFields condition key to control whether users can include optional metadata fields in their reports. For more information, see Control S3 Inventory report configuration creation.	February 20, 2024
IPv6 support for S3 on Outposts	You can now access S3 on Outposts buckets using IPv6 through S3 on Outposts dual-stack endpoints. IPv6 support for S3 on Outposts allows you to manage your S3 on Outposts buckets and control plane resources over IPv6 networks.	January 16, 2024
New high-performance, single-zone Amazon S3 storage class – S3 Express One Zone	Amazon S3 Express One Zone is a high-performance, single-zone Amazon S3 storage class that is purpose-built to deliver consistent, single-digit millisecond data access for your most latency-sensitive applications. For more information, see S3 Express One Zone.	November 28, 2023
Mountpoint for Amazon S3 adds support for S3 Express One Zone	You can now mount S3 Express One Zone directory buckets with Mountpoint.	November 28, 2023

Lambda invocation schema version	Amazon S3 Batch Operations introduces a new Lambda invocation schema version for use with Batch Operations jobs that act on directory buckets. For more information, see Using Lambda and Amazon S3 batch operations with directory buckets.	November 28, 2023
Import action for directory buckets	Amazon S3 introduces the import action. Import is a streamlined method for creating Amazon S3 Batch Operations jobs to copy objects from general purpose buckets to directory buckets. For more information, see Importing objects into a directory bucket.	November 28, 2023
Manage S3 access with S3 Access Grants	Amazon S3 Access Grants enables you to manage data permissions at scale for AWS Identity and Access Management (IAM) principals in addition to directory identities from corporate directories such as Azure AD. You can now enforce least-privilege S3 permissions and easily scale those permissions based on your business needs. For more information, see Managing access with S3 Access Grants.	November 26, 2023

Mountpoint for Amazon S3 adds caching feature	With Mountpoint, you can now configure caching for repeatedly accessed data.	November 22, 2023
Enhanced Amazon S3 Batch Operations manifest generation	You can now direct Amazon S3 Batch Operations to generate a manifest automatically based on object filter criteria that you specify when you create your job. This option is available for batch replication jobs that you create in the Amazon S3 console, or for any job type that you create by using the AWS CLI, AWS SDKs, or Amazon S3 REST API. For more information, see Creating an Amazon S3 Batch Operations job.	November 22, 2023
Existing Amazon S3 buckets can now add Object Lock configurations	You can now enable Object Lock on existing Amazon S3 bucket. You may set legal holds and retention periods for new or existing buckets. For more information, see Using Object Lock.	November 20, 2023
S3 Storage Lens request metrics for prefixes	S3 Storage Lens introduces request metrics for prefixes within an Amazon S3 bucket. For more information, see Metrics categories.	November 17, 2023

Amazon S3 Storage Lens groups	S3 Storage Lens introduces Storage Lens groups, a custom defined filter for objects based on object metadata. For more information, see Working with Amazon S3 Storage Lens groups.	November 15, 2023
New IAM policy	S3 on Outposts introduces `AWSServiceRoleForS3nOutposts`, a service-linked role to help manage network resources for you. For more information, see Using service-linked roles for S3 on Outposts.	October 3, 2023
Amazon S3 provides the `Last-Modified` time for delete markers	Amazon S3 provides the `Last-Modified` time of delete markers in the response headers of S3 `Head` and `Get` API operations. For more information, see Working with delete markers.	September 27, 2023
Amazon S3 update to AWS managed policy	Amazon S3 added `s3:Describe*` permissions to `AmazonS3ReadOnlyAccess`. For more information, see AWS managed policies for Amazon S3.	August 11, 2023

Improved start times for Standard restore requests made through S3 Batch Operations	Standard retrievals for restore requests that are made through S3 Batch Operations now can start within minutes. For more information, see Archive Retrieval Options.	August 9, 2023
Added Mountpoint, a high-throughput client for mounting an Amazon S3 bucket as a local file system.	With Mountpoint, your applications can access objects stored in Amazon S3 through file operations, g iving your applications access to the elastic storage and throughput of Amazon S3 t hrough a file interface.	August 9, 2023
Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)	Dual-layer server-side encryption with AWS Key Management Service (AWS KMS) keys (DSSE-KMS) ap plies two layers of encryptio n to objects when they are uploaded to Amazon S3. For more information, see Using dual-layer server-sid e encryption with AWS KMS keys.	June 13, 2023

Amazon S3 enables S3 Block Public Access and disables S3 access control lists (ACLs) for all new buckets.	Amazon S3 now automatically enables S3 Block Public Access and disables S3 access control lists (ACLs) for all new S3 buckets in all AWS Regions. For more information, see Blocking public access to your Amazon S3 storage and Controlling ownership of objects and disabling ACLs for your bucket.	April 27, 2023
S3 Replication Operations Failed metric	Amazon S3 adds new Amazon CloudWatch metric to monitor S3 Replication failures. For more information, see Monitoring progress with replication metrics.	April 5, 2023
Private DNS	AWS PrivateLink for Amazon S3 now supports Private DNS. For more information, see Private DNS.	March 14, 2023
Cross-account access points support in the Amazon S3 console	Amazon S3 now supports creating cross-account access points with the Amazon S3 console. For more information, see Creating access points.	March 14, 2023

Amazon S3 on Outposts supports S3 Replication on Outposts	With local S3 Replication, you can automatically replicate objects to a single Outposts destination bucket or to multiple destination buckets. The destination buckets can be in different AWS Outposts or within the same Outposts as the source bucket. For more information, see Replicating objects for S3 on Outposts.	March 14, 2023
Amazon S3 Object Lambda Access Point alias	When you create an Object Lambda Access Point, Amazon S3 automatically generates a unique alias for your Object Lambda Access Point. You can use this alias instead of an Amazon S3 bucket name or the Object Lambda Access Point Amazon Resource Name (ARN) in a request for access point data plane operations. For more information, see How to use a bucket-style alias for your Object Lambda Access Point.	March 14, 2023
Amazon S3 Multi-Region Access Points cross-account support	Amazon S3 now supports creating cross-account Multi-Region Access Points with the Amazon S3 console. For more information, see Creating Multi-Region Access Points.	March 14, 2023

Cross-account access points	Amazon S3 supports creating cross-account access points. You can create a cross-account access point by using the AWS Command Line Interface (AWS CLI) or the REST API `CreateAccessPoint` operation. For more information, see Creating access points.	November 30, 2022
Amazon S3 supports failover controls for Amazon S3 Multi-Region Access Points	Amazon S3 introduces failover control for Multi-Region Access Points. These controls let you shift S3 data access request traffic routed through an Amazon S3 Multi-Region Access Point to an alternate AWS Region within minutes to test and build highly available applications. For more information, see Amazon S3 Multi-Region Access Point failover controls.	November 28, 2022
Amazon S3 Storage Lens increases organization-wide visibility with 34 new metrics	S3 Storage Lens introduces 34 additional metrics to uncover deeper cost-optimization opportunities, identify data-protection best practices, and improve the performance of application workflows. For more information, see S3 Storage Lens metrics.	November 17, 2022

Amazon S3 supports higher restore request rates for S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive	Amazon S3 supports restore requests at a rate of up to 1,000 transactions per second, per AWS account for the S3 Glacier Flexible Retrieval and S3 Glacier Deep Archive storage classes.	November 15, 2022
Amazon S3 on Outposts supports additional S3 Lifecycle actions and filters	S3 on Outposts supports additional S3 Lifecycle rules to optimize capacity mana gement. You can expire objects as they age or are replaced with newer version s. You can create a lifecycle rule for a whole bucket or a subset of objects in a bucket by filtering with prefixes, object tags, or object size. For more information, see Creating and managing a lifecycle configuration.	November 2, 2022
S3 Replication support for SSE-C objects	You can replicate objects that are created using server-side encryption with customer -provided keys. For more information about replicati ng encrypted objects, see Replicating objects created with server-side encryption (SSE-C, SSE-S3, SSE-KMS).	October 24, 2022

Amazon S3 on Outposts supports access point aliases	With S3 on Outposts, you must use access points to access any object in an Outposts bucket. Every time you create an access point for a bucket, S3 on Outposts automatically generates an access point alias. You can use this access point alias instead of an access point ARN for any data plane operation. For more information, see Using a bucket-style alias for your S3 on Outposts bucket access point.	October 21, 2022
S3 Object Lambda supports the `HeadObject`, `ListObjects`, and `ListObjectsV2` operations	You can use custom code to modify the data returned by standard S3 `GET`, `LIST`, or `HEAD` requests to filter rows, dynamically resize images, redact confidential data, and more. For more information, see Transforming objects with S3 Object Lambda.	October 4, 2022

Amazon S3 on Outposts supports S3 Versioning	When enabled, S3 Versioning saves multiple distinct copies of an object in the same bucket. You can use S3 Versioning to preserve, retrieve, and restore every version of every object stored in your Outposts buckets. S3 Versioning helps you recover from unintended user actions and application failures. For more information, see Managing S3 Versioning for your S3 on Outposts bucket.	September 21, 2022
AWS Backup for Amazon S3	AWS Backup is a fully managed, policy-based service that you can use to define a central backup policy to protect your Amazon S3 data. For more information, see Using AWS Backup for Amazon S3.	February 18, 2022

Use S3 Batch Replication to replicate existing objects	With S3 Batch Replication, you can replicate objects that existed before a replication configuration was in place. Replicating existing objects is done through the use of a Batch Operations job. S3 Batch Replication differs from live replication, which continuously and automatic ally copies new objects across Amazon S3 buckets. For more information, see Replicating existing objects with S3 Batch Replication.	February 8, 2022
Rename of S3 Glacier Flexible Retrieval	The Glacier storage class has been renamed to S3 Glacier Flexible Retrieval. This change does not impact the API.	November 30, 2021

New S3 Object Ownership setting to disable ACLs	You can apply the bucket owner enforced setting for Object Ownership to disable ACLs for your bucket and the objects in it and take ownership of every object in your bucket. The bucket owner enforced setting simplifies access management for data stored in Amazon S3. For more information, see Controlling ownership of objects and disabling ACLs for your bucket.	November 30, 2021
New S3 Intelligent-Tiering storage class	S3 Intelligent-Tiering Archive Instant Access is an additional storage class under S3 Intelligent-Tiering. For more information, see How S3 Intelligent-Tiering works.	November 30, 2021
New S3 Glacier Instant Retrieval storage class	You can now place objects in the S3 Glacier Instant Retrieval storage class. For more information about this storage class, see Using Amazon S3 storage classes.	November 30, 2021

AWS Backup for Amazon S3 Preview	AWS Backup is a fully managed, policy-based service that you can use to define a central backup policy to protect your Amazon S3 data. For more informati on, see Using AWS Backup for Amazon S3.	November 30, 2021
AWS Identity and Access Management Access Analyzer for Amazon S3	IAM Access Analyzer runs policy checks to validate your policy against IAM policy grammar and best practices. To learn more about validatin g policies using IAM Access Analyzer, see IAM Access Analyzer policy validation in the *IAM User Guide*.	November 30, 2021
New event types	New event types added to Amazon S3 Event Notificat ions, see Amazon S3 Event Notifications.	November 29, 2021
Enable Amazon EventBridge on buckets	You can enable EventBrid ge on Amazon S3 buckets to send events to Amazon EventBridge, see Using EventBridge.	November 29, 2021
New S3 Lifecycle filters	You can create lifecycle rules based on object size or specify how many noncurre nt object versions to keep. For more information, see Examples of S3 Lifecycle configuration.	November 23, 2021

Publish Amazon S3 Storage Lens metrics to Amazon CloudWatch	You can publish S3 Storage Lens usage and activity metrics to Amazon CloudWatch to create a unified view of your operational health in CloudWatch dashboards. You can also use CloudWatch features, like alarms and triggered actions, metric math, and anomaly detection, to monitor and take action on S3 Storage Lens metrics. In addition, the CloudWatch APIs enable applications, including third-party providers, to access your S3 Storage Lens metrics. For more information, see the Monitor S3 Storage Lens metrics in CloudWatch.	November 22, 2021
Multi-Region Access Points	You can use Multi-Region Access Points to create a global endpoint that applications can use to fulfill requests from Amazon S3 buckets located in multiple AWS Regions. You can use this Multi-Region Access Point to route data to a bucket with the lowest latency. For more information about Multi-Region Access Points and how to use them, see Multi-Region Access Point in Amazon S3.	September 2, 2021

Amazon S3 on Outposts adds direct local access for applications	Run your applications outside the AWS Outposts virtual private cloud (VPC) and access your S3 on Outposts data. You can also access S3 on Outposts objects directly from your on-premises network. For more information about configuring S3 on Outposts endpoints using customer-owned IP (CoIP) addresses and accessing your objects by creating a local gateway from your on-premises network, see Accessing Amazon S3 on Outposts using VPC-only access points.	July 29, 2021
Amazon S3 access point alias	When you create an access point, Amazon S3 automatically generates an alias that you can use instead of a bucket name for data access. You can use this access point alias instead of an Amazon Resource Name (ARN) for any access point data plane operation. For more information, see Using a bucket-style alias for your access point.	July 26, 2021

Amazon S3 Inventory and S3 Batch Operations support S3 Bucket Key status	Amazon S3 Inventory and Batch Operations support identifying and copying existing objects with S3 Bucket Keys. S3 Bucket Keys accelerate the reduction of server-side encryption costs for existing objects. For more information, see Amazon S3 Inventory and Batch Operations Copy object.	June 3, 2021
Amazon S3 Storage Lens metrics account snapshot	The S3 Storage Lens account snapshot displays your total storage, object count, and average object size on the S3 console home (**Buckets**) page by summarizing metrics from your default dashboard. For more information, see S3 Storage Lens metrics account snapshot.	May 5, 2021
Increased Amazon S3 on Outposts endpoint support	S3 on Outposts now supports up to 100 endpoints per Outpost. For more information, see S3 on Outposts network restrictions.	April 29, 2021

Amazon S3 on Outposts event notifications in Amazon CloudWatch Events	You can use CloudWatch Events to create a rule to capture any S3 on Outposts API event and get notified through all supported CloudWatch targets. For more information, see Receiving S3 on Outposts event notifications using CloudWatch Events.	April 19, 2021
S3 Object Lambda	With S3 Object Lambda, you can add your own code to Amazon S3 GET requests to modify and process data as it is returned to an application. You can use custom code to modify the data returned by standard S3 GET requests to f ilter rows, dynamically resize images, redact confidential data, and more. For more information, see Transform ing objects.	March 18, 2021

AWS PrivateLink	With AWS PrivateLink for Amazon S3, you can connect directly to S3 by using an interface endpoint in your virtual private cloud (VPC) instead of connecting over the internet. Interface endpoints are directly accessible from applications that are on premises or in a different AWS Region. For more information, see AWS PrivateLink for Amazon S3.	February 2, 2021
Managing Amazon S3 on Outposts capacity with AWS CloudTrail	S3 on Outposts management events are available through CloudTrail logs. For more information, see Managing S3 on Outposts capacity with CloudTrail.	December 21, 2020
Strong consistency	Amazon S3 provides strong read-after-write consistency for PUT and DELETE requests of objects in your S3 bucket in all AWS Regions. In addition, read operations on Amazon S3 Select, Amazon S3 access control lists, Amazon S3 Object Tags, and object metadata (for example, HEAD object) are strongly consistent. For more information, see Amazon S3 data consistency model.	December 1, 2020

Amazon S3 replica modification sync	Amazon S3 replica modification sync keeps object metadata, such as tags, ACLs, and Object Lock settings, in sync between source objects and replicas. When this feature is enabled, Amazon S3 replicates metadata changes made to either the source object or the replica copies. For more information, see Replicating metadata changes with replica modification sync.	December 1, 2020
Amazon S3 Bucket Keys	Amazon S3 Bucket Keys reduce the cost of Amazon S3 server-side encryption with AWS Key Management Service (SSE-KMS). This new bucket-level key for server-side encryption can reduce AWS KMS request costs by up to 99 percent by decreasing the request traffic from Amazon S3 to AWS KMS. For more information, see Reducing the cost of SSE-KMS by using S3 Bucket Keys.	December 1, 2020

Amazon S3 Storage Lens	S3 Storage Lens aggregates your metrics and displays the information in the **Account snapshot** section on the Amazon S3 console **Buckets** page. S3 Storage Lens also provides an interactive dashboard that you can use to visualize insights and trends, flag outliers, and receive recommendations for optimizing storage costs and applying data protection best practices. Your dashboard has drill-down options to generate and visualize insights at the organization, account, AWS Region, storage class, bucket, prefix, or Storage Lens group level. You can also send a daily metrics report in CSV or Parquet format to a general purpose S3 bucket or export the metrics directly to an AWS-managed S3 table bucket. For more information, see Assessing your storage activity and usage with S3 Storage Lens.	November 18, 2020

Tracing S3 requests using AWS X-Ray	Amazon S3 integrates with X-Ray to propagate the trace context and give you one request chain with upstream and downstream nodes. For more information, see Tracing requests using X-Ray.	November 16, 2020
S3 Replication metrics	S3 Replication metrics provide detailed metrics for the replication rules in your replication configuration. For more information, see Replication metrics and Amazon S3 event notifications.	November 9, 2020
S3 Intelligent-Tiering Archive Access and Deep Archive Access	S3 Intelligent-Tiering Archive Access and Deep Archive Access are additional storage tiers under S3 Intelligent-Tiering. For more information, see Storage class for automatically optimizing frequently and infrequently accessed objects .	November 9, 2020
Delete marker replication	With delete marker replication, you can ensure that delete markers are copied to your destination buckets for your replication rules. For more information, see Using delete marker replication.	November 9, 2020

S3 Object Ownership	Object Ownership is an S3 bucket setting that you can use to control ownership of new objects that are uploaded to your buckets. For more information, see Using S3 Object Ownership.	October 2, 2020
Amazon S3 on Outposts	With Amazon S3 on Outposts, you can create S3 buckets on your AWS Outposts resources and easily store and retrieve objects on-premises for applications that require local data access, local data processing, and data residency. You can use S3 on Outposts through the AWS Management Console, AWS CLI, AWS SDKs, or REST API. For more information, see Using Amazon S3 on Outposts.	September 30, 2020
Bucket owner condition	You can use the Amazon S3 bucket owner condition to ensure that the buckets you use in your S3 operations belong to the AWS accounts that you expect. For more information, see Bucket owner condition.	September 11, 2020

S3 Batch Operations support for Object Lock Retention	You can now use Batch Operations with S3 Object Lock to apply retention settings to many Amazon S3 objects at once. For more information, see Setting S3 Object Lock Retention dates with S3 Batch Operations.	May 4, 2020
S3 Batch Operations support for Object Lock Legal Hold	You can now use Batch Operations with S3 Object Lock to add a legal hold to many Amazon S3 objects at once. For more information, see Using S3 Batch Operations for setting S3 Object Lock Legal Hold.	May 4, 2020
Job tags for S3 Batch Operations	You can add tags to your S3 Batch Operations jobs to control and label those jobs. For more information, see Tags for S3 Batch Operations jobs.	March 16, 2020
Amazon S3 access points	Amazon S3 access points simplify managing data access at scale for shared datasets in S3. Access points are named network endpoints that are attached to buckets that you can use to perform S3 object operations. For more information, see Managing data access with Amazon S3 access points.	December 2, 2019

Access Analyzer for Amazon S3	Access Analyzer for Amazon S3 alerts you to S3 buckets that are configured to allow access to anyone on the internet or other AWS accounts, including accounts outside of your organization. For more information, see Using Access Analyzer for Amazon S3.	December 2, 2019
S3 Replication Time Control (S3 RTC)	S3 Replication Time Control (S3 RTC) replicates most objects that you upload to Amazon S3 in seconds, and 99.99 percent of those objects within 15 minutes. For more information, see Replicating objects using S3 Replication Time Control (S3 RTC).	November 20, 2019
Same-Region Replication	You can use Same-Region Replication (SRR) to copy objects across Amazon S3 buckets in the same AWS Region. For information about both Cross-Region Replication (CRR) and Same-Region Replication, see Replication.	September 18, 2019
Cross-Region Replication support for S3 Object Lock	Cross-Region Replication now supports Object Lock. For more information, see What does Amazon S3 Replicate?.	May 28, 2019

S3 Batch Operations	By using S3 Batch Operations, you can perform large-scale Batch Operations on Amazon S3 objects. S3 Batch Operations can run a single operation on lists of objects that you specify. A single job can perform the specified operation on billions of objects containing exabytes of data. For more information, see Performing S3 Batch Operations.	April 30, 2019
Asia Pacific (Hong Kong) Region	Amazon S3 is now available in the Asia Pacific (Hong Kong) Region. For more information about Amazon S3 Regions and endpoints, see Regions and endpoints in the *AWS General Reference*.	April 24, 2019
Added a new field to the server access logs	Amazon S3 added the following new field to the server access logs: Transport Layer Security (TLS) version. For more information, see Server access log format.	March 28, 2019
New archive storage class	Amazon S3 now offers a new archive storage class, S3 Glacier Deep Archive (`DEEP_ARCHIVE`), for storing rarely accessed objects. For more information, see Storage Classes.	March 27, 2019

Added new fields to the server access logs	Amazon S3 added the following new fields to the server access logs: Host Id, Signature Version, Cipher Suite, Authentication Type, and Host Header. For more information, see Server access log format.	March 5, 2019
Support for Parquet-formatted Amazon S3 Inventory files	Amazon S3 now supports the Apache Parquet (Parquet) format in addition to the Apache optimized row columnar (ORC) and comma-separated values (CSV) file formats for inventory output files. For more information, see Inventory.	December 4, 2018
S3 Object Lock	Amazon S3 now offers Object Lock functionality that provides Write Once Read Many (WORM) protections for Amazon S3 objects. For more information, see Locking Objects.	November 26, 2018
Restore speed upgrade	Using Amazon S3 restore speed upgrade, you can change the speed of a restoration from the S3 Glacier Flexible Retrieval storage class to a faster speed while the restoration is in progress. For more information, see Restoring Archived Objects.	November 26, 2018

Restore Event Notifications	Amazon S3 Event Notifications now support initiation and completion events when restoring objects from the S3 Glacier Flexible Retrieval storage class. For more information, see Event Notifications.	November 26, 2018

PUT directly to the S3 Glacier Flexible Retrieval storage class	The Amazon S3 PUT operation now supports specifying S3 Glacier Flexible Retrieval as the storage class when creating objects. Previously, you had to transition objects to the S3 Glacier Flexible Retrieval storage class from another Amazon S3 storage class. Also, when using S3 Cross-Region Replication (CRR), you can now specify S3 Glacier Flexible Retrieval as the storage class for replicated objects. For more information about the S3 Glacier Flexible Retrieval storage class, see Storage Classes. For more information about specifying the storage class for replicated objects, Replication Configuration Overview. For more informati on about the direct PUT to S3 Glacier Flexible Retrieval REST API changes, see Document History: PUT directly to S3 Glacier Flexible Retrieval .	November 26, 2018

New storage class	Amazon S3 now offers a new storage class named S3 Intelligent-Tiering (`INTELLIGENT_TIERING`) that is designed for long-lived data with changing or unknown access patterns. For more information, see Storage Classes.	November 26, 2018
Amazon S3 Block Public Access	Amazon S3 now includes the ability to block public access to buckets and objects on a per-bucket or account-wide basis. For more information, see Using Amazon S3 Block Public Access.	November 15, 2018
Filtering enhancements in Cross-Region Replication (CRR) rules	In a CRR rule configuration, you can specify an object filter to choose a subset of objects to apply the rule to. Previously, you could filter only on an object key prefix. In this release, you can filter on an object key prefix, one or more object tags, or both. For more information, see CRR Setup: Replication Configuration Overview.	September 19, 2018

New Amazon S3 Select features	Amazon S3 Select now supports Apache Parquet input, queries on nested JSON objects, and two new Amazon CloudWatch monitoring metrics (`SelectScannedBytes` and `SelectReturnedBytes`).	September 5, 2018
Updates now available over RSS	You can now subscribe to an RSS feed to receive notifications about updates to the *Amazon S3 User Guide*.	June 19, 2018

Earlier updates

The following table describes the important changes in each release of the *Amazon S3 User Guide* before June 19, 2018.

Change	Description	Date
Code examples update	Code examples updated: • C#—Updated all of the examples to use the task-based asynchronous pattern. For more information, see Amazon Web Services Asynchronous APIs for .NET in the *AWS SDK for .NET Developer Guide*. Code examples are now compliant with version 3 of the AWS SDK for .NET. • Java—Updated all of the examples to use the client builder model. For more information about the client builder model, see Creating Service Clients. •	April 30, 2018

Change	Description	Date
	PHP—Updated all of the examples to use the AWS SDK for PHP 3.0. For more information about the AWS SDK for PHP 3.0, see AWS SDK for PHP. • Ruby—Updated example code so that the examples work with the AWS SDK for Ruby version 3.	

Amazon S3 now reports S3 Glacier Flexible Retrieval and ONEZONE_IA storage classes to Amazon CloudWatch Logs storage metrics	In addition to reporting actual bytes, these storage metrics include per-object overhead bytes for applicable storage classes (ONEZONE_IA , STANDARD_IA , and S3 Glacier Flexible Retrieval):
	• For ONEZONE_IA and STANDARD_IA storage class objects, Amazon S3 reports objects smaller than 128 KB as 128 KB. For more information, see Understanding and managing Amazon S3 storage classes.
	• For S3 Glacier Flexible Retrieval storage class objects, the storage metrics report the following overheads:
	  • A 32 KB per-object overhead, charged at S3 Glacier Flexible Retrieval storage class pricing
	  • An 8 KB per-object overhead, charged at STANDARD storage class pricing
	  For more information, see Transitioning objects using Amazon S3 Lifecycle.
	For more information about storage metrics, see Monitoring metrics with Amazon CloudWatch.
		April 30, 2018

Change	Description	Date
New storage class	Amazon S3 now offers a new storage class, `STANDARD_IA` (IA, for infrequent access) for storing objects. This storage class is optimized for long-lived and less frequently accessed data. For more information, see Understanding and managing Amazon S3 storage classes.	April 4, 2018
Amazon S3 Select	a>Amazon S3 now supports retrieving object content based on an SQL expression. For more information, see Querying data in place with Amazon S3 Select.	April 4, 2018
Asia Pacific (Osaka-Local) Region	Amazon S3 is now available in the Asia Pacific (Osaka-Local) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*. **Important** You can use the Asia Pacific (Osaka-Local) Region only in conjunction with the Asia Pacific (Tokyo) Region. To request access to Asia Pacific (Osaka-Local) Region, contact your sales representative.	February 12, 2018
Amazon S3 Inventory creation timestamp	Amazon S3 Inventory now includes a timestamp of the date and start time of the creation of the Amazon S3 Inventory report. You can use the timestamp to determine changes in your Amazon S3 storage from the start time of when the inventory report was generated.	January 16, 2018
Europe (Paris) Region	Amazon S3 is now available in the Europe (Paris) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.	December 18, 2017

Change	Description	Date
China (Ningxia) Region	Amazon S3 is now available in the China (Ningxia) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.	November 29, 2017
Support for ORC-formatted Amazon S3 Inventory files	Amazon S3 now supports the Apache optimized row columnar (ORC) format in addition to comma-separated values (CSV) file format for inventory output files. Also, you can now query Amazon S3 inventory using standard SQL by using Amazon Athena, Amazon Redshift Spectrum, and other tools such as Presto, Apache Hive, and Apache Spark. For more information, see Cataloging and analyzing your data with S3 Inventory.	November 17, 2017
Default encryption for S3 buckets	Amazon S3 default encryption provides a way to set the default encryption behavior for an S3 bucket. You can set default encryption on a bucket so that all objects are encrypted when they are stored in the bucket. The objects are encrypted using server-side encryption with either Amazon S3 managed keys (SSE-S3) or AWS managed keys (SSE-KMS). For more information, see Setting default server-side encryption behavior for Amazon S3 buckets.	November 06, 2017
Encryption status in Amazon S3 Inventory	Amazon S3 now supports including encryption status in Amazon S3 Inventory so you can see how your objects are encrypted at rest for compliance auditing or other purposes. You can also configure to encrypt Amazon S3 Inventory with server-side encryption (SSE) or SSE-KMS so that all inventory files are encrypted accordingly. For more information, see Cataloging and analyzing your data with S3 Inventory.	November 06, 2017

Change	Description	Date
Cross-Region Replication (CRR) enhancements	Cross-Region Replication now supports the following: • In a cross-account scenario, you can add a CRR configuration to change replica ownership to the AWS account that owns the destination bucket. For more information, see Changing the replica owner. • By default, Amazon S3 does not replicate objects in your source bucket that are created using server-side encryption using keys stored in AWS KMS In your CRR configuration, you can now direct Amazon S3 to replicate these objects. For more information, see Replicating encrypted objects (SSE-S3, SSE-KMS, DSSE-KMS, SSE-C).	November 06, 2017
Europe (London) Region	Amazon S3 is now available in the Europe (London) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.	December 13, 2016
Canada (Central) Region	Amazon S3 is now available in the Canada (Central) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.	December 8, 2016

Change	Description	Date
Object tagging	Amazon S3 now supports object tagging. Object tagging enables you to categorize storage. Object key name prefixes also enable you to categorize storage, object tagging adds another dimension to it. There are added benefits tagging offers. These include: • Object tags enable fine-grained access control of permissions (for example, you could grant an IAM user permissions to read-only objects with specific tags). • Fine-grained control in specifying lifecycle configuration. You can specify tags to select a subset of objects to which lifecycle rule applies. • If you have Cross-Region Replication (CRR) configured, Amazon S3 can replicate the tags. You must grant necessary permission to the IAM role created for Amazon S3 to assume to replicate objects on your behalf. • You can also customize CloudWatch metrics and CloudTrail events to display information by specific tag filters. For more information, see Tagging your objects.	November 29, 2016
Amazon S3 Lifecycle now supports tag-based filters	Amazon S3 now supports tag-based filtering in lifecycle configuration. You can now specify lifecycle rules in which you can specify a key prefix, one or more object tags, or a combination of both to select a subset of objects to which the lifecycle rule applies. For more information, see Managing the lifecycle of objects.	November 29, 2016

Change	Description	Date
CloudWatch request metrics for buckets	Amazon S3 now supports CloudWatch metrics for requests made on buckets. When you enable these metrics for a bucket, the metrics report at 1-minut e intervals. You can also configure which objects in a bucket will report these request metrics. For more information, see Monitoring metrics with Amazon CloudWatch.	November 29, 2016
Amazon S3 Inventory	Amazon S3 now supports storage inventory. Amazon S3 Inventory provides a flat-file output of your objects and their corresponding metadata on a daily or weekly basis for an S3 bucket or a shared prefix (that is, objects that have names that begin with a common string). For more information, see Cataloging and analyzing your data with S3 Inventory.	November 29, 2016
Amazon S3 Analytics – Storage Class Analysis	The new Amazon S3 analytics – storage class analysis feature observes data access patterns to help you determine when to transition less frequently accessed `STANDARD` storage to the `STANDARD_IA` (IA, for infrequent access) storage class. After storage class analysis observes the infrequent access patterns of a filtered set of data over a period of time, you can use the analysis results to help you improve your lifecycle configurations. This feature also includes a detailed daily analysis of your storage usage at the specified bucket, pr efix, or tag level that you can export to an S3 bucket.	November 29, 2016
New Expedited and Bulk data retrieval s when restoring archived objects from Amazon Glacier	Amazon S3 now supports Expedited and Bulk data retrievals in addition to Standard retrievals when restoring objects archived to Amazon Glacier. For more information, see Restoring an archived object.	November 21, 2016

Change	Description	Date
CloudTrail object logging	CloudTrail supports logging Amazon S3 object level API operations such as `GetObject` , `PutObject` , and `DeleteObject` . You can configure your event selectors to log object level API operations. For more information, see Logging Amazon S3 API calls using AWS CloudTrail.	November 21, 2016
US East (Ohio) Region	Amazon S3 is now available in the US East (Ohio) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.	October 17, 2016
IPv6 support for Amazon S3 Transfer Acceleration	Amazon S3 now supports Internet Protocol version 6 (IPv6) for Amazon S3 Transfer Acceleration. You can connect to Amazon S3 over IPv6 by using the new dual-stack for Transfer Acceleration endpoint. For more information, see Getting started with Amazon S3 Transfer Acceleration.	October 6, 2016
IPv6 support	Amazon S3 now supports Internet Protocol version 6 (IPv6). You can access Amazon S3 over IPv6 by using dual-stack endpoints. For more information, see Making requests to Amazon S3 over IPv6 in the *Amazon S3 API Reference*.	August 11, 2016
Asia Pacific (Mumbai) Region	Amazon S3 is now available in the Asia Pacific (Mumbai) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.	June 27, 2016

Change	Description	Date
Amazon S3 Transfer Acceleration	Amazon S3 Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. Transfer Acceleration takes advantage of Amazon CloudFront globally distributed edge locations. For more information, see Configuring fast, secure file transfers using Amazon S3 Transfer Acceleration.	April 19, 2016
Lifecycle support to remove expired object delete markers	Lifecycle configuration `Expiration` action now allows you to direct Amazon S3 to remove expired object delete markers in a/ versioned bucket. For more information, see Elements to describe lifecycle actions.	March 16, 2016

Change	Description	Date
Bucket lifecycle configuration now supports action to stop incomplete multipart uploads	Bucket lifecycle configuration now supports the `AbortIncompleteMultipartUpload` action that you can use to direct Amazon S3 to stop multipart uploads that don't complete within a specified number of days after being initiated. When a multipart upload becomes eligible for a stop operation, Amazon S3 deletes any uploaded parts and stops the multipart upload. For conceptual information, see the following topics in the *Amazon S3 User Guide*: • Aborting a multipart upload • Elements to describe lifecycle actions The following API operations have been updated to support the new action: • PUT Bucket lifecycle – The XML configuration now allows you to specify the `AbortIncompleteMultipartUpload` action in a lifecycle configuration rule. • List Parts and Initiate Multipart Upload – Both of these API operations now return two additional response headers (`x-amz-abort-date` , and `x-amz-abort-rule-id`) if the bucket has a lifecycle rule that specifies the `AbortIncompleteMultipartUpload` action. These headers in the response indicate when the initiated multipart upload becomes eligible for a stop operation and which lifecycle rule is applicable.	March 16, 2016

Change	Description	Date
Asia Pacific (Seoul) Region	Amazon S3 is now available in the Asia Pacific (Seoul) Region. For more information about Amazon S3 Regions and endpoints, see Regions and Endpoints in the *AWS General Reference*.	January 6, 2016
New condition key and a multipart upload change	IAM policies now support an Amazon S3 `s3:x-amz-storage-class` condition key. For more information, see Bucket policy examples using condition keys. You no longer need to be the initiator of a multipart upload to upload parts and complete the upload. For more information, see Multipart upload API and permissions.	December 14, 2015
Renamed the US Standard Region	Changed the Region name string from "US Standard" to "US East (N. Virginia)." This is only a Region name update, there is no change in the functionality.	December 11, 2015
New storage class	Amazon S3 now offers a new storage class, STANDARD_IA (IA, for infrequent access) for storing objects. This storage class is optimized for long-lived and less frequently accessed data. For more information, see Understanding and managing Amazon S3 storage classes. Lifecycle configuration feature updates now allow you to transition objects to the STANDARD_IA storage class. For more information, see Managing the lifecycle of objects. Previously, the Cross-Region Replication feature used the storage class of the source object for object replicas. Now, when you configure Cross-Region Replication, you can specify a storage class for the object replica created in the destination bucket. For more information, see Replicating objects within and across Regions.	September 16, 2015

Change	Description	Date
AWS CloudTrail integration	New AWS CloudTrail integration allows you to record Amazon S3 API activity in your S3 bucket. You can use CloudTrail to track S3 bucket creations or deletions, access control modifications, or lifecycle configuration changes. For more information, see Logging Amazon S3 API calls using AWS CloudTrail.	September 1, 2015
Bucket limit increase	Amazon S3 now supports bucket limit increases. By default, customers can create up to 100 buckets in their AWS account. Customers who need additional buckets can increase that limit by submitting a service limit increase. For information about how to increase your bucket limit, go to AWS service quotas in the *AWS General Reference*. For more information, see Using the AWS SDKs and General purpose bucket quotas, limitations, and restrictions.	August 4, 2015
Consistency model update	Amazon S3 now supports read-after-write consistency for new objects added to Amazon S3 in the US East (N. Virginia) Region. Prior to this update, all Regions except US East (N. Virginia) Region supported read-after-write consistency for new objects uploaded to Amazon S3. With this enhancement, Amazon S3 now supports read-after-write consistency in all Regions for new objects added to Amazon S3. Read-after-write consistency allows you to retrieve objects immediately after creation in Amazon S3. For more information, see Regions.	August 4, 2015
Event notifications	Amazon S3 Event Notifications have been updated to add notifications when objects are deleted and to add filtering on object names with prefix and suffix matching. For more information, see Amazon S3 Event Notifications.	July 28, 2015

Change	Description	Date
Amazon CloudWatch integration	New Amazon CloudWatch integration allows you to monitor and set alarms on your Amazon S3 usage through CloudWatch metrics for Amazon S3. Supported metrics include total bytes for Standard storage, total bytes for Reduced-Redundancy Storage, and total number of objects for a given S3 bucket. For more information, see Monitoring metrics with Amazon CloudWatch.	July 28, 2015
Support for deleting and emptying non-empty buckets	Amazon S3 now supports deleting and emptying non-empty buckets. For more information, see Emptying a general purpose bucket.	July 16, 2015
Bucket policies for Amazon VPC endpoints	Amazon S3 has added support for bucket policies for virtual private cloud (VPC) (VPC) endpoints. You can use S3 bucket policies to control access to buckets from specific VPC endpoints, or specific VPCs. VPC endpoints are easy to configure, are highly reliable, and provide a secure connection to Amazon S3 without requiring a gateway or a NAT instance. For more information, see Controlling access from VPC endpoints with bucket policies.	April 29, 2015
Event notifications	Amazon S3 Event Notifications have been updated to support the switch to resource-based permissions for AWS Lambda functions. For more information, see Amazon S3 Event Notifications.	April 9, 2015
Cross-Region Replication	Amazon S3 now supports Cross-Region Replication. Cross-Region Replication is the automatic, asynchronous copying of objects across buckets in different AWS Regions. For more information, see Replicating objects within and across Regions.	March 24, 2015

Change	Description	Date
Event notifications	Amazon S3 now supports new event types and destinations in a bucket notification configuration. Prior to this release, Amazon S3 supported only the *s3:ReducedRedundancyLostObject* event type and an Amazon SNS topic as the destination. For more information about the new event types, see Amazon S3 Event Notifications.	November 13, 2014
Server-side encryption with customer-provided encryption keys	Server-side encryption with AWS Key Management Service (AWS KMS) keys (SSE-KMS) Amazon S3 now supports server-side encryption using AWS KMS. This feature allows you to manage the envelope key through AWS KMS, and Amazon S3 calls AWS KMS to access the envelope key within the permissions you set. For more information about server-side encryption with AWS KMS, see Protecting Data Using Server-Side Encryption with AWS Key Management Service.	November 12, 2014
Europe (Frankfurt) Region	Amazon S3 is now available in the Europe (Frankfurt) Region.	October 23, 2014
Server-side encryption with customer-provided encryption keys	Amazon S3 now supports server-side encryption using customer-provided encryption keys (SSE-C). Server-side encryption enables you to request Amazon S3 to encrypt your data at rest. When using SSE-C, Amazon S3 encrypts your objects with the custom encryption keys that you provide. Since Amazon S3 performs the encryption for you, you get the benefits of using your own encryption keys without the cost of writing or executing your own encryption code. For more information about SSE-C, see Server-Side Encryption (Using Customer-Provided Encryption Keys).	June 12, 2014

Change	Description	Date
Lifecycle support for versioning	Prior to this release, lifecycle configuration was supported only on nonversioned buckets. Now you can configure lifecycle on both nonversioned and versioning-enabled buckets. For more information, see Managing the lifecycle of objects.	May 20, 2014
Access control topics revised	Revised Amazon S3 access control documentation. For more information, see Identity and Access Management for Amazon S3.	April 15, 2014
Server access logging topic revised	Revised server access logging documentation. For more information, see Logging requests with server access logging.	November 26, 2013
.NET SDK samples updated to version 2.0	.NET SDK samples in this guide are now compliant to version 2.0.	November 26, 2013
SOAP Support Over HTTP deprecated	SOAP support over HTTP is deprecated, but it is still available over HTTPS. New Amazon S3 features will not be supported for SOAP. We recommend that you use either the REST API or the AWS SDKs.	September 20, 2013
IAM policy variable support	IAM policy language now supports variables. When a policy is evaluated, any policy variables are replaced with values that are supplied by context-based information from the authenticated user's session. You can use policy variables to define general purpose policies without explicitly listing all the components of the policy. For more information about policy variables, see IAM Policy Variables Overview in the *IAM User Guide*. For examples of policy variables in Amazon S3, see Identity-based policy examples for Amazon S3.	April 3, 2013

Change	Description	Date
Console support for Requester Pays	You can now configure your bucket for Requester Pays by using the Amazon S3 console. For more information, see Using Requester Pays general purpose buckets for storage transfers and usage.	December 31, 2012
Root domain support for website hosting	Amazon S3 now supports hosting static websites at the root domain. Visitors to your website can access your site from their browser without specifying **www** in the web address (for example, they can use **example.com** instead of **www.example.com**). Many customers already host static websites on Amazon S3 that are accessible from a **www** subdomain (for example, **www.example.com**). Previously, to support root domain access, you needed to run your own web server to proxy root domain requests from browsers to your website on Amazon S3. Running a web server to proxy requests introduces additional costs, operational burden, and another potential point of failure. Now, you can take advantage of the high availability and durability of Amazon S3 for both **www** and root domain addresses. For more information, see Hosting a static website using Amazon S3.	December 27, 2012
Console revision	Amazon S3 console has been updated. The documentation topics that refer to the console have been revised accordingly.	December 14, 2012

Change	Description	Date
Support for Archiving Data to Amazon Glacier	Amazon S3 now supports a storage option that enables you to utilize Amazon Glacier's low-cost storage service for data archival. To archive objects, you define archival rules identifying objects and a time frame when you want Amazon S3 to archive these objects to Amazon Glacier. You can easily set the rules on a bucket using the Amazon S3 console or programmatically using the Amazon S3 API or AWS SDKs. For more information, see Managing the lifecycle of objects.	November 13, 2012
Support for Website Page Redirects	For a bucket that is configured as a website, Amazon S3 now supports redirecting a request for an object to another object in the same bucket or to an external URL. For more information, see (Optional) Configuring a webpage redirect. For information about hosting websites, see Hosting a static website using Amazon S3.	October 4, 2012
Support for Cross-Origin Resource Sharing (CORS)	Amazon S3 now supports Cross-Origin Resource Sharing (CORS). CORS defines a way in which client web applications that are loaded in one domain can interact with or access resources in a different domain. With CORS support in Amazon S3, you can build rich client-side web applications on top of Amazon S3 and selectively allow cross-domain access to your Amazon S3 resources. For more information, see Using cross-origin resource sharing (CORS).	August 31, 2012
Support for Cost Allocation Tags	Amazon S3 now supports cost allocation tagging, which allows you to label S3 buckets so you can more easily track their cost against projects or other criteria. For more information about using tagging for buckets, see Using cost allocation S3 bucket tags.	August 21, 2012

Change	Description	Date
Support for MFA-protected API access in bucket policies	Amazon S3 now supports MFA-protected API access, a feature that can enforce AWS Multi-Factor Authentication for an extra level of security when accessing your Amazon S3 resources. It is a security feature that requires users to prove physical possession of an MFA device by providing a valid MFA code. For more information, go to AWS Multi-Factor Authentication. You can now require MFA authentication for any requests to access your Amazon S3 resources. To enforce MFA authentication, Amazon S3 now supports the `aws:MultiFactorAuthAge` key in a bucket policy. For an example bucket policy, see Requiring MFA.	July 10, 2012
Object Expiration support	You can use Object Expiration to schedule automatic removal of data after a configured time period. You set object expiration by adding lifecycle configuration to a bucket.	27 December 2011
New Region supported	Amazon S3 now supports the South America (São Paulo) Region. For more information, see Accessing an Amazon S3 general purpose bucket.	December 14, 2011
Multi-Object Delete	Amazon S3 now supports Multi-Object Delete API that enables you to delete multiple objects in a single request. With this feature, you can remove large numbers of objects from Amazon S3 more quickly than using multiple individual DELETE requests. For more information, see Deleting Amazon S3 objects.	December 7, 2011
New Region supported	Amazon S3 now supports the US West (Oregon) Region. For more information, see Buckets and Regions.	November 8, 2011
Documentation Update	Documentation bug fixes.	November 8, 2011

Change	Description	Date
Documentation Update	In addition to documentation bug fixes, this release includes the following enhancements: • New server-side encryption sections using the AWS SDK for PHP and the AWS SDK for Ruby (see Specifying server-side encryption with Amazon S3 managed keys (SSE-S3)).	October 17, 2011
Server-side encryption support	Amazon S3 now supports server-side encryption. It enables you to request Amazon S3 to encrypt your data at rest, that is, encrypt your object data when Amazon S3 writes your data to disks in its data centers. In addition to REST API updates, the AWS SDK for Java and .NET provide necessary functionality to request server-side encryption. You can also request server-side encryption when uploading objects using the AWS Management Console. To learn more about data encryption, go to Using Data Encryption.	October 4, 2011
Documentation Update	In addition to documentation bug fixes, this release includes the following enhancements: • Added Ruby and PHP samples to the Making requests in the *Amazon S3 API Reference* section. • Added sections describing how to generate and use presigned URLs. For more information, see Sharing objects with presigned URLs and Sharing objects with presigned URLs. • Updated an existing section to introduce AWS Explorers for Eclipse and Visual Studio. For more information, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.	September 22, 2011

Change	Description	Date
Support for sending requests using temporary security credentials	In addition to using your AWS account and IAM user security credentials to send authenticated requests to Amazon S3, you can now send requests using temporary security credentials you obtain from AWS Identity and Access Management (IAM). You can use the AWS Security Token Service API or the AWS SDK wrapper libraries to request these temporary credentials from IAM. You can request these temporary security credentials for your own use or hand them out to federated users and applications. This feature enables you to manage your users outside AWS and provide them with temporary security credentials to access your AWS resources. For more information, see Making requests in the *Amazon S3 API Reference*. For more information about IAM support for temporary security credentials, see Temporary Security Credentials in the *IAM User Guide*.	August 3, 2011
Multipart Upload API extended to enable copying objects up to 5 TB	Prior to this release, Amazon S3 API supported copying objects of up to 5 GB in size. To enable copying objects larger than 5 GB, Amazon S3 now extends the multipart upload API with a new operation, `Upload Part (Copy)`. You can use this multipart upload operation to copy objects up to 5 TB in size. For more information, see Copying, moving, and renaming objects. For conceptual information about multipart upload API, see Uploading and copying objects using multipart upload in Amazon S3.	June 21, 2011
SOAP API calls over HTTP disabled	To increase security, SOAP API calls over HTTP are disabled. Authenticated and anonymous SOAP requests must be sent to Amazon S3 using SSL.	June 6, 2011

Change	Description	Date
IAM enables cross-account delegation	Previously, to access an Amazon S3 resource, an IAM user needed permissions from both the parent AWS account and the Amazon S3 resource owner. With cross-account access, the IAM user now only needs permission from the owner account. That is, If a resource owner grants access to an AWS account, the AWS account can now grant its IAM users access to these resources. For more information, see Creating a role to delegate permissions to an IAM user in the *IAM User Guide*. For more information on specifying principals in a bucket policy, see Principals for bucket policies.	June 6, 2011
New link	This service's endpoint information is now located in the *AWS General Reference*. For more information, go to Regions and Endpoints in the AWS General Reference.	March 1, 2011
Support for hosting static websites in Amazon S3	Amazon S3 introduces enhanced support for hosting static websites. This includes support for index documents and custom error documents. When using these features, requests to the root of your bucket or a subfolder (for example, `http://mywebsite.com/subfolder`) returns your index document instead of the list of objects in your bucket. If an error is encountered, Amazon S3 returns your custom error message instead of an Amazon S3 error message. For more information, see Hosting a static website using Amazon S3.	June 6, 2011

Change	Description	Date
This service's endpoint information is now located in the *AWS General Reference*. For more information, go to Regions and Endpoints in the AWS General Reference.	March 1, 2011	
Support for hosting static websites in Amazon S3	Amazon S3 introduces enhanced support for hosting static websites. This includes support for index documents and custom error documents. When using these features, requests to the root of your bucket or a subfolder (for example, `http://mywebsite.com/subfolder`) returns your index document instead of the list of objects in your bucket. If an error is encountered, Amazon S3 returns your custom error message instead of an Amazon S3 error message. For more information, see Hosting a static website using Amazon S3.	February 17, 2011
Response Header API Support	The GET Object REST API now allows you to change the response headers of the REST GET Object request for each request. That is, you can alter object metadata in the response, without altering the object itself. For more information, see Downloading objects.	January 14, 2011
Large object support	Amazon S3 has increased the maximum size of an object you can store in an S3 bucket from 5 GB to 5 TB. If you are using the REST API, you can upload objects of up to 5 GB in a single PUT operation. For larger objects, you must use the Multipart Upload REST API to upload objects in parts. For more information, see Uploading and copying objects using multipart upload in Amazon S3.	December 9, 2010

Change	Description	Date
Multipart upload	Multipart upload enables faster, more flexible uploads into Amazon S3. It allows you to upload a single object as a set of parts. For more information, see Uploading and copying objects using multipart upload in Amazon S3.	November 10, 2010
Canonical ID support in bucket policies	You can now specify canonical IDs in bucket policies. For more information, see Principals for bucket policies	September 17, 2010
Amazon S3 works with IAM	This service now integrates with AWS Identity and Access Management (IAM). For more information, go to AWS services that work with IAM in the *IAM User Guide*.	September 2, 2010
Notifications	The Amazon S3 notifications feature enables you to configure a bucket so that Amazon S3 publishes a message to an Amazon Simple Notification Service (Amazon SNS) topic when Amazon S3 detects a key event on a bucket. For more information, see Setting Up Notification of Bucket Events.	July 14, 2010
Bucket policies	Bucket policies are an access management system that you use to set access permissions across buckets, objects, and sets of objects. This functionality supplements and in many cases replaces access control lists. For more information, see Bucket policies for Amazon S3.	July 6, 2010
Path-style syntax available in all Regions	Amazon S3 now supports the path-style syntax for any bucket in the US Classic Region, or if the bucket is in the same Region as the endpoint of the request. For more information, see Virtual Hosting.	June 9, 2010
New endpoint for Europe (Ireland)	Amazon S3 now provides an endpoint for Europe (Ireland) : `http://s3-eu-west-1.amazonaws.com` .	June 9, 2010

Change	Description	Date
Console	You can now use Amazon S3 through the AWS Management Console. You can read about all of the Amazon S3 functionality in the console in the Amazon Simple Storage Service User Guide.	June 9, 2010
Reduced Redundancy	Amazon S3 now enables you to reduce your storage costs by storing objects in Amazon S3 with reduced redundancy. For more information, see Reduced Redundancy Storage.	May 12, 2010
New Region supported	Amazon S3 now supports the Asia Pacific (Singapore) Region. For more information, see Buckets and Regions.	April 28, 2010
Object Versioning	This release introduces object versioning. All objects now can have a key and a version. If you enable versioning for a bucket, Amazon S3 gives all objects added to a bucket a unique version ID. This feature enables you to recover from unintended overwrites and deletions. For more information, see Versioning and Using Versioning.	February 8, 2010
New Region supported	Amazon S3 now supports the US West (N. California) Region. The new endpoint for requests to this Region is `s3-us-west-1.amazonaws.com`. For more information, see Buckets and Regions.	December 2, 2009
AWS SDK for .NET	AWS now provides libraries, sample code, tutorials, and other resources for software developers who prefer to build applications using .NET language-specific API operations instead of REST or SOAP. These libraries provide basic functions (not included in the REST or SOAP APIs), such as request authentication, request retries, and error handling so that it's easier to get started. For more information about language-specific libraries and resources, see Developing with Amazon S3 using the AWS SDKs in the *Amazon S3 API Reference*.	November 11, 2009